



# Panduan *Secure* SDLC

Oleh : Restia Moegiono {CEH|CHFI|ECSA|QRMO}

TLP : CLEAR

Dokumen ini bisa disebarluaskan secara bebas

## Riwayat Perubahan

Versi	Tanggal	Personel	Perubahan
0.0	25 Maret 2023	Restia Moegiono	Versi awal Panduan <i>Secure</i> SDLC
1.0	3 November 2023	Restia Moegiono	Perbaikan penulisan

# Daftar Isi

Riwayat Perubahan.....	i
Daftar Isi.....	ii
<b>Bab I - Pendahuluan .....</b>	<b>1</b>
1.1    Latar Belakang.....	1
1.2    Tujuan Panduan <i>Secure</i> SLDC .....	1
1.3    Manfaat <i>Secure</i> SLDC .....	1
<b>Bab II - Persyaratan Keamanan (<i>Security Requirement</i>) .....</b>	<b>3</b>
2.1    Persyaratan Keamanan Inti ( <i>Core Security Requirement</i> ) .....	3
2.2.1    Identifikasi Persyaratan Keamanan Inti.....	3
a.    Persyaratan Kerahasiaan ( <i>Confidentiality</i> ) .....	3
b.    Persyaratan Integritas ( <i>Integrity</i> ) .....	5
c.    Persyaratan Ketersediaan ( <i>Availability</i> ) .....	5
d.    Persyaratan Autentikasi .....	5
e.    Persyaratan Otorisasi.....	8
f.    Persyaratan Akuntabilitas .....	9
2.2.2    Identifikasi Persyaratan Umum.....	9
a.    Persyaratan Manajemen Sesi.....	9
b.    Persyaratan Manajemen <i>Error</i> dan <i>Exception</i> .....	9
c.    Persyaratan Manajemen Parameter Konfigurasi .....	10
2.2.3    Identifikasi Persyaratan Operasional .....	10
a.    Persyaratan Lingkungan <i>Deployment</i> .....	10
b.    Persyaratan Pengarsipan .....	10
c.    Persyaratan Anti-Pembajakan ( <i>Anti-Piracy</i> ) .....	10
2.2.4    Identifikasi Persyaratan Lain .....	10
a.    Persyaratan Urutan dan Waktu .....	10
b.    Persyaratan Internasional.....	10
c.    Persyaratan Pengadaan .....	10
2.2    Klasifikasi Data.....	11
2.2.1    Tipe Data .....	11
a.    Data Terstruktur .....	11
b.    Data Tidak Terstruktur .....	11
2.2.2    Memberi Label pada Data .....	11
2.2.3    Kepemilikan dan Peran pada Data .....	11
2.2.4 <i>Data Lifecycle Management</i> (DLM) .....	12
2.2.5    Persyaratan Privasi .....	12
a.    Anonimisasi Data .....	12
b.    Penghapusan Data .....	12

c.	<i>Security Model</i> .....	13
d.	Pseudonimisasi ( <i>Pseudonymization</i> ).....	13
2.3	Pemodelan <i>Use Case</i> dan <i>Misuse Case</i> .....	13
2.3.1	Menganalisis Skenario <i>Use Case</i> .....	14
2.3.2	Menganalisis Skenario <i>Misuse Case</i> .....	14
2.3.3	Membuat <i>Attack Model</i> .....	15
2.3.4	Memilih Kontrol Mitigasi.....	15
2.4	Manajemen Risiko.....	15
2.4.1	Penilaian Risiko.....	15
a.	Langkah 1: Karakterisasi Sistem.....	16
b.	Langkah 2: Identifikasi Ancaman.....	16
c.	Langkah 3: Identifikasi Kerentanan.....	16
d.	Langkah 4: Analisis Kontrol.....	16
e.	Langkah 5: Penentuan Tingkat Kemungkinan.....	16
f.	Langkah 6: Penentuan Tingkat Dampak.....	16
g.	Langkah 7: Penentuan Tingkat Risiko.....	17
h.	Langkah 8: Rekomendasi Kontrol.....	17
i.	Langkah 9: Dokumentasi Hasil.....	17
2.4.2	Mitigasi Risiko.....	17
a.	Opsi Mitigasi Risiko.....	17
b.	Strategi Mitigasi Risiko.....	18
c.	Pendekatan pada Implementasi Kontrol Keamanan.....	18
d.	Kategori Kontrol Keamanan.....	19
e.	Analisis Biaya dan Manfaat.....	19
f.	Risiko Residual.....	19
2.4.3	Evaluasi dan Penilaian Risiko.....	19
<b>Bab III</b>	<b>- Desain Keamanan (<i>Security Design</i>).....</b>	<b>20</b>
3.1	Desain Keamanan Inti ( <i>Core Security Design</i> ).....	20
3.1.1	Desain Kerahasiaan.....	20
3.1.2	Desain Integritas.....	22
3.1.3	Desain Ketersediaan.....	23
3.1.4	Desain Autentikasi.....	24
3.1.5	Desain Otorisasi.....	24
a.	Direktori.....	24
b.	<i>Access Control List (ACL)</i> .....	25
c.	Matriks Kontrol Akses ( <i>Access Control Matrix</i> ).....	25
d.	Kapabilitas ( <i>Capability</i> ).....	26
e.	Kontrol Akses Berorientasi Prosedur ( <i>Procedure Oriented Access Control</i> ).....	26
3.1.6	Desain Akuntabilitas.....	26
3.2	Desain Tambahan.....	26
3.2.1	Bahasa Pemrograman.....	26
a.	Kode Tidak Terkelola ( <i>Unmanaged Code</i> ).....	26
b.	Kode terkelola ( <i>Managed code</i> ).....	27
3.2.2	Jenis, Format, Jangkauan, dan Panjang Data.....	27

a.	Tipe Data Primitif atau <i>Built-In</i> .....	27
b.	Tipe Data yang Ditentukan oleh Pemrogram ( <i>User-Defined Data Types</i> ) .....	27
c.	Nilai dan Operasi yang Diizinkan ( <i>Set of Values and Permissible Operations</i> ) .....	27
d.	Ketidacocokan dan kesalahan konversi ( <i>Conversion Mismatches and Casting or Conversion Errors</i> ) .....	28
3.2.3	Keamanan <i>Database</i> .....	28
a.	<i>Polyinstantiation</i> .....	29
b.	Enkripsi <i>Database</i> ( <i>Database Encryption</i> ).....	29
c.	Normalisasi ( <i>Normalization</i> ) .....	30
d.	<i>Trigger</i> dan <i>View</i> .....	30
3.2.4	Desain Antarmuka .....	30
a.	Antarmuka Pengguna ( <i>User Interface/UI</i> ) .....	30
b.	Antarmuka Pemrograman Aplikasi ( <i>Application Programming Interface/API</i> ).....	31
c.	Antarmuka Manajemen Keamanan ( <i>Security Management Interface/SMI</i> ).....	31
d.	Antarmuka <i>Out-of-Band</i> .....	32
e.	Antarmuka <i>Log</i> .....	32
3.2.5	Interkonektivitas .....	33
3.3	<i>Threat Modeling</i> .....	33
3.3.1	Langkah 1: Dekomposisi <i>Software</i> .....	33
3.3.2	Langkah 2: Menentukan dan Mengurutkan Ancaman .....	34
3.3.3	Langkah 3: Menentukan Penanggulangan dan Mitigasi .....	34
<b>Bab IV</b>	<b>- Pengembangan Keamanan (<i>Security Development</i>) .....</b>	<b>35</b>
4.1	Identifikasi Kerentanan dan Penerapan Kontrol Umum pada <i>Software</i> .....	35
4.1.1	Penerapan <i>Database</i> Kerentanan ( <i>Vulnerability Database</i> ).....	35
4.1.2	Praktek <i>Secure Coding</i> .....	35
4.2	Proses Pengembangan <i>Software</i> yang Aman .....	36
4.2.1.	<i>Source Code Versioning</i> .....	36
4.2.2.	<i>Code Analysis</i> .....	36
a.	<i>Static Code Analysis</i> .....	36
b.	<i>Dynamic Code Analysis</i> .....	37
4.2.3.	<i>Code Review</i> .....	37
4.2.4.	Pengujian oleh <i>Pengembang</i> ( <i>Developer Testing</i> ).....	37
a.	Pengujian Unit ( <i>Unit Test</i> ) .....	38
b.	Pengujian Integrasi ( <i>Integration Test</i> ) .....	38
c.	Pengujian Regresi ( <i>Regression Test</i> ).....	38
d.	Pengujian <i>Software</i> .....	38
4.3	Mengamankan Lingkungan Pengembangan .....	38
4.3.1.	Mengamankan Akses Fisik ke <i>Software</i> yang Membangun <i>Code</i> .....	38
4.3.2.	Menggunakan <i>Access Control List</i> ( <i>ACL</i> ) .....	38
4.3.3.	Menggunakan <i>Software</i> untuk <i>Version Control</i> .....	38
4.3.4.	<i>Build Automation</i> .....	39
4.3.5.	Penandatanganan <i>Code</i> ( <i>Code Signing</i> ) .....	39
<b>Bab V</b>	<b>- Pengujian Keamanan (<i>Security Testing</i>).....</b>	<b>40</b>

5.1	Validasi <i>Attack Surface</i> .....	40
5.1.1	Pengujian Pasca Pengembangan <i>Software</i> .....	40
5.1.2	Melakukan Pengujian Keamanan .....	40
a.	Pengujian <i>White Box</i> ( <i>White Box Testing</i> ).....	40
b.	Pengujian <i>Black Box</i> ( <i>Black Box Testing</i> ).....	40
c.	Pengujian Validasi Kriptografi ( <i>Cryptographic Validation Testing</i> ) .....	40
5.1.3	Melakukan Pengujian Keamanan <i>Software</i> untuk <i>Quality Assurance</i> .....	40
5.2	Manajemen Data Pengujian.....	41
5.2.1	Identifikasi <i>Output</i> Data Pengujian.....	41
5.2.2	Melakukan Pengujian dengan Transaksi Sintetis.....	41
5.2.3	<i>Tool</i> pada Manajemen Data Pengujian .....	41
5.2.4	Pelaporan dan Pelacakan Cacat.....	42
<b>Bab VI - Penerapan Keamanan (<i>Security Deployment</i>).....</b>		<b>43</b>
6.1	Pertimbangan Penerimaan <i>Software</i> .....	43
6.1.1	Kriteria Penyelesaian <i>Software</i> .....	43
6.1.2	Manajemen Perubahan.....	43
6.1.3	Persetujuan Manajemen untuk <i>Deployment</i> atau Rilis <i>Software</i> .....	43
6.1.4	Kebijakan Penerimaan dan Pengecualian Risiko.....	44
6.1.5	Dokumentasi <i>Software</i> .....	44
6.2	Verifikasi dan Validasi (V&V) .....	44
6.2.1	Reviu.....	44
6.2.2	Pengujian .....	44
a.	Pengujian Deteksi Kesalahan ( <i>Error Detection Test</i> ) .....	44
b.	Pengujian Penerimaan ( <i>Acceptance Test</i> ) .....	44
c.	Pengujian Pihak Independen .....	44
6.3	Sertifikasi dan Akreditasi (C&A) .....	45
6.3.1	Sertifikasi <i>Software</i> .....	45
6.3.2	Akreditasi <i>Software</i> .....	45
6.4	Instalasi .....	45
6.4.1	<i>Hardening</i> .....	45
6.4.2	Konfigurasi Lingkungan <i>Production</i> .....	45
6.4.3	Manajemen Rilis.....	46
6.4.4	Bootstrap dan <i>Startup</i> yang Aman .....	46
<b>Bab VII - Pemeliharaan Keamanan (<i>Security Maintenance</i>) .....</b>		<b>47</b>
7.1	Operasional, Pemantauan dan Pemeliharaan .....	47
7.1.1	Pengamanan Operasional <i>Software</i> .....	47
7.1.2	Pemantauan Berkelanjutan .....	47
7.1.3	Audit untuk Pemantauan .....	48
7.2	Manajemen Insiden Siber.....	48
7.2.1	Menentukan <i>Event</i> , <i>Alert</i> , dan Insiden Siber .....	48
7.2.2	Identifikasi Jenis Insiden Siber .....	48
7.2.3	Proses Tanggap Insiden Siber .....	48
7.3	Manajemen Permasalahan .....	49

7.3.1	Pemberitahuan Insiden Siber.....	49
7.3.2	Analisis Akar Penyebab ( <i>Root Cause Analysis</i> ).....	49
7.3.3	Penentuan Solusi.....	49
7.3.4	Permintaan Perubahan .....	49
7.3.5	Menerapkan Solusi.....	49
7.3.6	Memantau dan Melaporkan .....	50
7.4	Manajemen Perubahan .....	50
7.4.1	Manajemen <i>Patch</i> dan Kerentanan .....	50
7.4.2	Pencadangan, Pemulihan, dan Pengarsipan.....	51
7.5	Penghapusan .....	51
7.5.1	Kebijakan Akhir Masa Pakai ( <i>End-of-Life</i> ) .....	51
7.5.2	Kriteria Pembuangan ( <i>Sunset Criteria</i> ) .....	51
7.5.3	Proses Penghapusan ( <i>Sunset Process</i> ) .....	51
7.5.4	Penghapusan Informasi dan Sanitasi Media.....	52
<b>Checklist Secure SDLC .....</b>		<b>53</b>
A.	<i>Checklist</i> Persyaratan Keamanan ( <i>Security Requirement</i> ).....	53
B.	<i>Checklist</i> Desain Keamanan ( <i>Security Design</i> ) .....	56
C.	<i>Checklist</i> Pengembangan Keamanan ( <i>Security Development</i> ) .....	58
D.	<i>Checklist</i> Pengujian Keamanan ( <i>Security Testing</i> ) .....	59
E.	<i>Checklist</i> Penerapan Keamanan ( <i>Security Deployment</i> ) .....	60
F.	<i>Checklist</i> Pemeliharaan Keamanan ( <i>Security Maintenance</i> ).....	62
<b>Referensi.....</b>		<b>64</b>

# Bab I - Pendahuluan

## 1.1 Latar Belakang

Keamanan siber menjadi hal yang sangat penting karena banyak hal yang saat ini tergantung pada penggunaan komputer dan internet, serta meningkatnya tren ancaman siber yang terjadi belakangan ini. Dimana *root cause* dari banyak insiden siber yang terjadi adalah kerentanan *software* dan kompleksitas *software* itu sendiri. Oleh karena itu, kerentanan *software* sudah seharusnya dicegah dan ini adalah sebagai tanggung jawab semua pihak.

*Secure Software Development Life Cycle* (*Secure SDLC*) berupaya menjadikan pengembangan *software* yang aman sejak awal. Dan penerapan *secure SDLC* merupakan hal yang penting karena keamanan dan integritas *software* merupakan hal yang penting. Hal ini dapat mengurangi risiko kerentanan *software* pada fase dalam *production*, serta meminimalkan dampaknya jika ditemukan. *Secure SDLC* menempatkan keamanan di depan dan di tengah, tidak hanya di belakang melalui kegiatan *vulnerability assessment* dan *penetration testing*. *Secure SDLC* adalah evolusi dari proses SDLC klasik yang mengintegrasikan keamanan di semua tahap pengembangan *software*, memastikan bahwa semua tim yang terlibat mempertimbangkan persyaratan fungsional proyek dan aspek keamanannya.

Adapun perbandingan antara SDLC klasik dan *secure SDLC*, yaitu:

SDLC Klasik	<i>Secure SDLC</i>
Fokusnya adalah mengembangkan aplikasi yang efisien dan produktif dengan biaya minimum dan secepat mungkin	Fokusnya adalah pada pengembangan aplikasi yang aman tanpa berdampak pada biaya, waktu pengiriman, dan efisiensi
Pengujian keamanan dan pengkodean aman tidak termasuk dalam tahapan prosesnya	Pengujian keamanan dan pengkodean yang aman adalah bagian mendasar dari proses ini. Pengujian dilakukan menjelang akhir proses
Pengujian dilakukan menjelang akhir proses	Pengujian dimulai pada tahap awal dan berlanjut sepanjang keseluruhan proses
Keamanan adalah sebuah renungan	Keamanan disertakan pada setiap tahap siklus hidup ( <i>life cycle</i> )

## 1.2 Tujuan Panduan *Secure SDLC*

Tujuan pembuatan Panduan *Secure SDLC* ini, antara lain:

1. Memastikan pengembangan *software* dilakukan secara aman.
2. Menjadi panduan bagi pengembang *software* dalam menerapkan *Secure SDLC*.

## 1.3 Manfaat *Secure SDLC*

Manfaat dari menerapkan *secure SDLC*, antara lain:

1. Mengurangi biaya pengembangan secara keseluruhan



Hal ini terjadi karena mengidentifikasi dan memperbaiki kerentanan dan kesalahan di awal proses, maka tidak perlu lagi melakukan *patch* pada *software* setelah fase *production*. *National Institute for Standards and Technology* (NIST) telah menunjukkan dalam sebuah penelitian bahwa organisasi harus mengeluarkan biaya 5 (lima) kali lebih banyak ketika perbaikan diterapkan setelah *software* dirilis.

2. Meningkatkan kolaborasi

*Secure SDLC* membangun budaya yang berfokus pada keamanan, menciptakan lingkungan kerja yang mengutamakan keamanan, dan semua pihak yang berpartisipasi dalam proyek bertanggung jawab atas keamanan *software* (tim pengembang, tim operasional, tim manajemen, dan *stakeholder* lainnya). Semua *stakeholder* ini bekerja sama untuk mencapai tujuan yang sama, yaitu membangun *software* yang aman dan berkualitas tinggi yang tidak mengorbankan kenyamanan demi keamanan (atau sebaliknya). Sehingga perbaikan terjadi di seluruh bagian organisasi dan aspek keamanan tidak lagi menjadi beban, melainkan ini menjadi tanggung jawab bersama.

3. Mendukung pengembangan yang cepat

Memasukkan pengujian keamanan di setiap tahapan SDLC (bukan hanya di akhir) tidak akan memperlambat proses pengembangan *software*, namun akan meningkatkannya. Mengapa? Karena waktu yang mungkin hilang dalam pengujian terus-menerus akan secara signifikan mengurangi jumlah modifikasi dan perbaikan di menit-menit terakhir yang memakan waktu.

4. Memperbaiki masalah dengan lebih cepat dan dengan lebih sedikit kesalahan

Kerentanan yang diidentifikasi pada tahap awal pengembangan, akan dapat diperbaiki oleh pengembang yang sama yang membuat *code* pada *software* tersebut. Ini jauh lebih baik daripada meminta tim lain (yang tidak berpartisipasi dalam proses pengembangan) untuk memperbaikinya nanti. Hal ini tidak hanya mempercepat proses pengembangan, tetapi juga mengurangi kemungkinan kesalahan.

5. Melindungi seluruh organisasi dari serangan siber

Setelah proses *secure SDLC* dilakukan, maka postur keamanan meningkat di seluruh bagian organisasi. Organisasi yang sadar akan keamanan dapat mengurangi risiko serangan siber secara signifikan. Karena risiko dari pelanggaran keamanan (*security breach*) sangat besar. Sebagai contoh, jika penyerang berhasil membobol *software* maka akan dengan mudah mendapatkan akses ke seluruh jaringan organisasi. Dan organisasi berisiko mengalami kerugian dalam banyak hal, seperti hilangnya kepercayaan *stakeholder*, menurunnya angka pendapatan, ketidakpatuhan terhadap regulasi, adanya tuntutan hukum, sampai dengan gagalnya organisasi mencapai misi dan visi organisasi. Dampak kerugian yang ditimbulkan bisa sangat besar dan jangkauannya luas.

6. Membantu organisasi mematuhi Undang-Undang Pelindungan Data Pribadi

Dengan memasukkan keamanan dalam tahap perencanaan dan desain akan bermanfaat bagi organisasi untuk memastikan semua elemen yang diwajibkan dalam Undang-Undang Pelindungan Data Pribadi sudah diterapkan. Hal ini tentu lebih baik daripada membayar denda atau mendapatkan sanksi akibat kelalaian dalam menjaga data pribadi pelanggan.

## Bab II - Persyaratan Keamanan (*Security Requirement*)

Tahap persyaratan keamanan ini bertujuan untuk:

- Mengidentifikasi dan menentukan persyaratan keamanan yang relevan untuk *software* yang dikembangkan.
- Mengidentifikasi persiapan yang dibutuhkan dalam aspek keamanan dalam pengembangan *software*.

### 2.1 Persyaratan Keamanan Inti (*Core Security Requirement*)

Tahap persyaratan keamanan inti bertujuan untuk:

- Mendefinisikan dan menangani tujuan keamanan informasi organisasi secara eksplisit.
- Mengidentifikasi persyaratan yang dapat diterapkan pada konteks bisnis dan fungsionalitas *software* yang melayani konteks tersebut.

#### 2.2.1 Identifikasi Persyaratan Keamanan Inti

##### a. Persyaratan Kerahasiaan (*Confidentiality*)

Persyaratan kerahasiaan bertujuan untuk memberikan perlindungan terhadap pengungkapan informasi/data yang bersifat rahasia/sensitif secara tidak sah. Persyaratan kerahasiaan harus ditentukan sepanjang siklus hidup informasi, mulai dari data dibuat hingga data dihapus:

- Data dalam transmisi (*data in transit*), yaitu ketika data ditransmisikan melalui jaringan dari satu titik ke titik lainnya.
- Data dalam pemrosesan (*data in process*), yaitu ketika data disimpan dalam memori komputer atau media untuk diproses dan diakses oleh pengguna.
- Data dalam penyimpanan (*data at rest*), yaitu ketika data tidak aktif, dalam sistem transaksional maupun sistem non-transaksional, termasuk arsip (data tidak aktif). Data ini disimpan atau dilestarikan.

Adapun mekanisme perlindungan kerahasiaan adalah sebagai berikut:

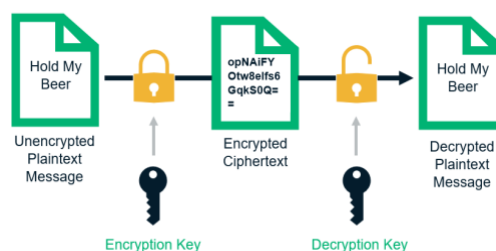
##### 1) Kriptografi

Kriptografi adalah mekanisme perlindungan yang bertujuan untuk mencegah pengungkapan informasi yang dianggap rahasia, menggunakan mekanisme:

##### a) Mekanisme terbuka (*overt*)

Tujuan dari mekanisme terbuka ini adalah untuk membuat informasi tidak dapat dibaca/dipahami, bahkan jika informasi diungkapkan sekalipun. Contohnya enkripsi dan fungsi *hash*.

#### How Encryption Works



Contoh enkripsi.

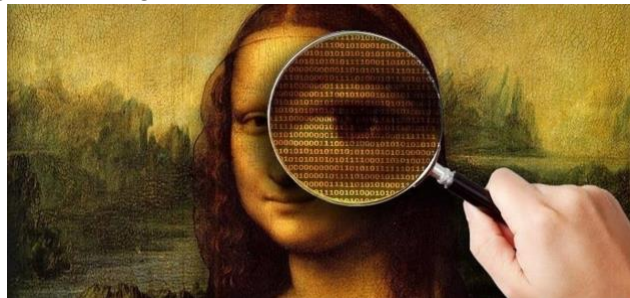
## An Example of a Hash Function



Contoh fungsi *hash*.

### b) Mekanisme rahasia (*covert*)

Tujuan dari mekanisme rahasia ini adalah untuk menyembunyikan informasi di dalam suatu media atau bentuk yang lain. Contohnya steganografi dan *digital watermark*.



Contoh steganografi.

A.

B.

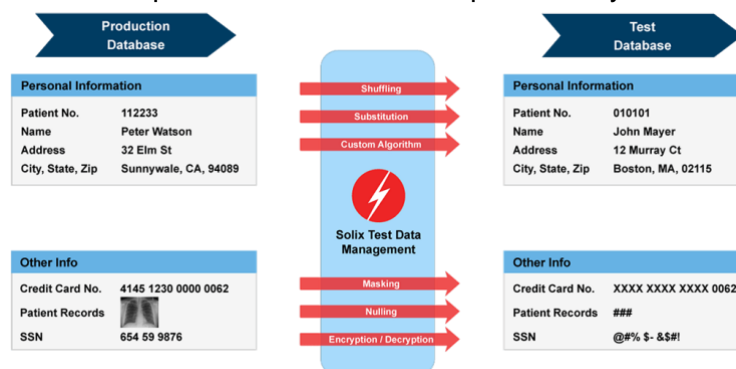
C.



Contoh *digital watermark*.

### 2) Penyamaran (*masking*)

Penyamaran adalah mekanisme perlindungan yang lebih lemah dari mekanisme perlindungan kriptografi, dimana informasi asli diberi tanda bintang (X). Mekanisme penyamaran digunakan untuk melindungi dari serangan selancar bahu (*shoulder surfing attack*), yaitu serangan dimana seseorang melihat dari balik bahu orang lain dan mengamati informasi sensitif. Contohnya: penyamaran nomor kartu kredit kecuali 4 (empat) digit terakhir saat dicetak pada kuitansi atau ditampilkan di layar.



Contoh *data masking*.

### b. Persyaratan Integritas (*Integrity*)

Persyaratan integritas bertujuan untuk mengatasi 2 (dua) aspek utama keamanan *software*, yaitu jaminan keandalan dan perlindungan/pencegahan terhadap modifikasi yang tidak sah. Integritas mengacu tidak hanya pada perlindungan modifikasi *software* (integritas sistem) tetapi juga data yang ditangani oleh *software* (integritas data).

Keandalan pada dasarnya bertujuan untuk memastikan bahwa *software* berfungsi seperti yang dirancang dan diharapkan. Ini juga dimaksudkan untuk memberikan kontrol keamanan yang akan memastikan bahwa keakuratan *software* dan data tetap terjaga. Pelindungan integritas juga mempertimbangkan kelengkapan dan konsistensi *software* atau data yang ditangani *software*.

### c. Persyaratan Ketersediaan (*Availability*)

Persyaratan ketersediaan adalah persyaratan *software* yang memastikan tidak ada gangguan terhadap operasional bisnis dan memastikan terdapat pelindungan terhadap upaya penghancuran *software* dan/atau data, sehingga membantu pencegahan *Denial of Service* (DoS) bagi pengguna.

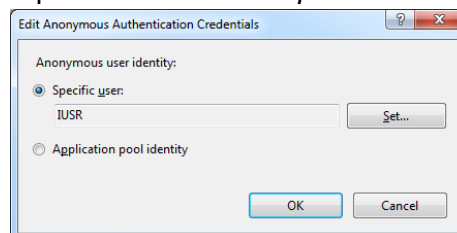
### d. Persyaratan Autentikasi

Persyaratan autentikasi bertujuan untuk memverifikasi dan memastikan keabsahan dan validitas identitas yang mengajukan klaim entitas. Kredensial autentikasi dapat diberikan oleh berbagai faktor atau kombinasi faktor yang mencakup pengetahuan (*what you know*), kepemilikan (*what you have*), atau karakteristik (*what you are*).

Bentuk autentikasi yang paling umum adalah sebagai berikut:

#### 1) Autentikasi Anonim (*Anonymous Authentication*)

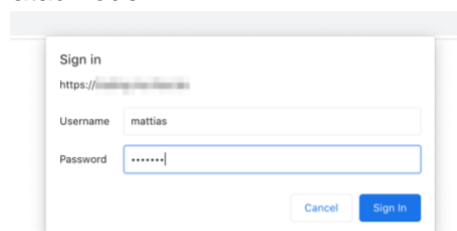
Autentikasi anonim adalah sarana akses ke area publik *software* tanpa meminta kredensial seperti *username* dan *password*.



Contoh autentikasi anonim.

#### 2) Autentikasi Dasar (*Basic Authentication*)

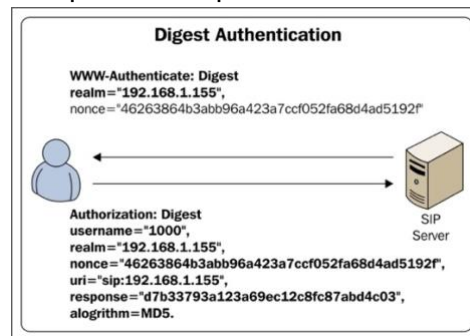
Autentikasi dasar adalah salah satu spesifikasi HTTP 1.0 yang dicirikan oleh *browser* yang meminta pengguna untuk memberikan kredensial dalam bentuk teks jelas (*cleartext*) atau kode.



Contoh autentikasi dasar.

### 3) Autentikasi Intisari (*Digest Authentication*)

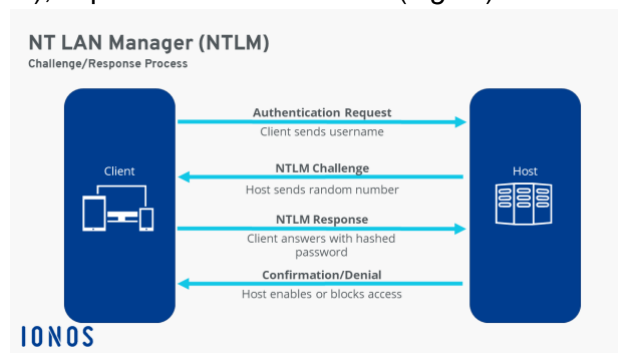
Autentikasi intisari (*digest*) adalah mekanisme *challenge response* yang mengirimkan *digest* dari pesan berupa nilai *hash* dari kredensial asli.



Contoh *digest authentication*.

### 4) Autentikasi Terintegrasi (*Integrated Authentication*)

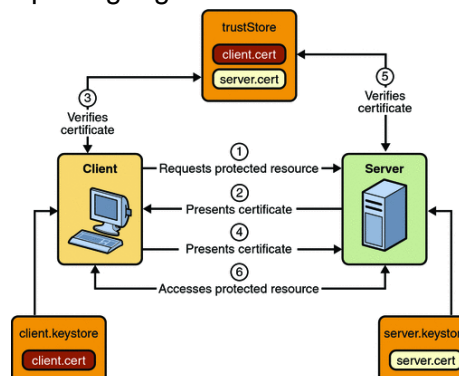
Autentikasi terintegrasi umumnya dikenal sebagai autentikasi New Technology LAN Manager (NTLM) atau autentikasi tantangan/respons New Technology (NT), seperti autentikasi intisari (*digest*).



Contoh autentikasi terintegrasi.

### 5) Autentikasi Berbasis Sertifikat Elektronik Milik *Client* (*Client Certificate-Based Authentication*)

Autentikasi ini memvalidasi identitas pemegang sertifikat elektronik (*client*). Sertifikat elektronik ini dikeluarkan oleh *Certification Authority* (CA) yang menjamin keabsahan pemegang sertifikat elektronik.

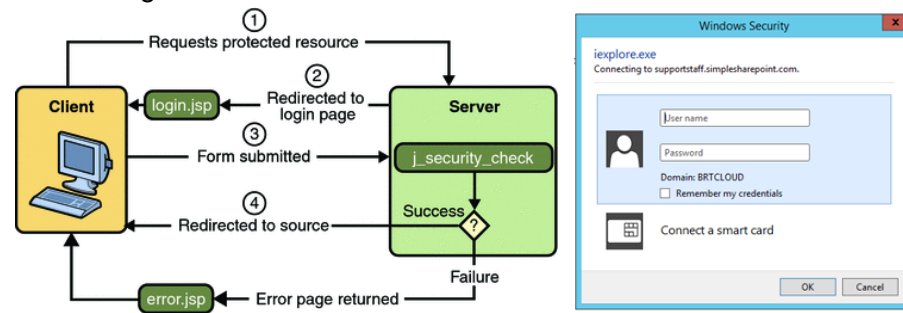


Contoh autentikasi berbasis sertifikat elektronik.

### 6) Autentikasi Formulir (*Form Authentication*)

Autentikasi formulir mengharuskan pengguna untuk memberikan *username* dan *password* untuk tujuan autentikasi, dan kredensial tersebut akan

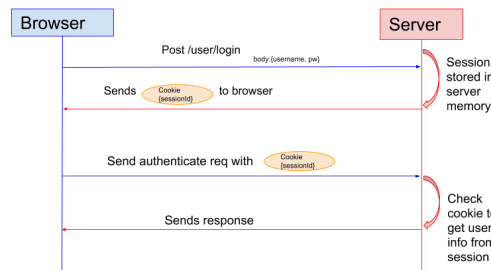
divalidasi terhadap direktori penyimpanan, seperti *active direktori*, *database*, atau *file* konfigurasi.



Contoh autentikasi formulir.

## 7) Autentikasi Berbasis Token (*Token-Based Authentication*)

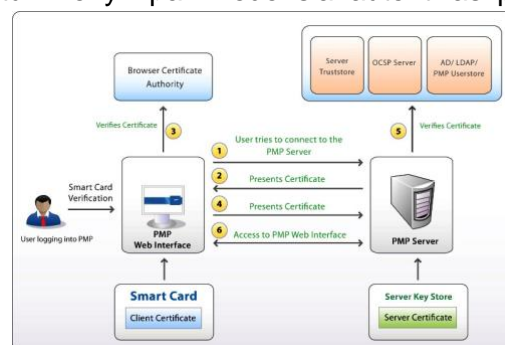
Autentikasi berbasis token biasanya digunakan bersama dengan autentikasi formulir dimana *username* dan *password* disediakan untuk verifikasi. Setelah verifikasi, token akan dikeluarkan untuk pengguna yang memberikan kredensial. Token kemudian digunakan untuk memberikan akses ke sumber daya yang diminta.



Contoh autentikasi berbasis token.

## 8) Autentikasi Berbasis Smart Card (*Smart Card-Based Authentication*)

Autentikasi berbasis *smart card* dilakukan berdasarkan validasi kepemilikan (*what you have*). *Smart card* berisi *microchip* tertanam yang dapat diprogram dan digunakan untuk menyimpan kredensial autentikasi pemilik *smart card*.



Contoh autentikasi berbasis *smart card*.

## 9) Autentikasi Biometrik

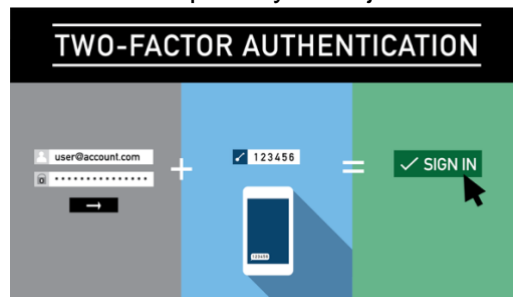
Autentikasi biometrik menggunakan karakteristik biologis (*what you are*) untuk memberikan kredensial identitas. Fitur biologis seperti pola pembuluh darah retina, fitur wajah, dan sidik jari digunakan untuk tujuan verifikasi identitas.



Contoh autentikasi biometrik.

#### 10) Autentikasi dua faktor (*two factor authentication/2FA*)

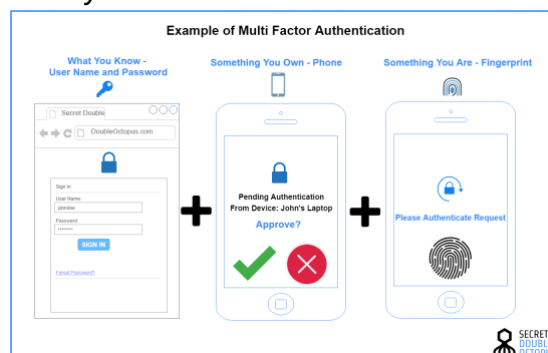
Autentikasi yang menggunakan 2 (dua) faktor untuk memvalidasi klaim dan/atau kredensial entitas, misalnya menggunakan faktor pertama berupa *username* dan *password* (*what you know*), kemudian menggunakan faktor kedua berupa *One Time Password* (OTP) yang dikirimkan melalui SMS atau menggunakan aplikasi autentikasi (*what you have*). Verifikasi dilakukan secara bersamaan setelah seluruh pertanyaan dijawab.



Contoh autentikasi dua faktor.

#### 11) Autentikasi multi faktor (*multi factor authentication/MFA*)

Autentikasi yang menggunakan lebih dari 2 (dua) faktor untuk memvalidasi klaim dan/atau kredensial entitas, yaitu kombinasi dari aspek *what you know*, *what you have*, dan *what you are*.



Contoh dari autentikasi multi faktor.

#### e. Persyaratan Otorisasi

Persyaratan otorisasi bertujuan untuk mengkonfirmasi bahwa entitas yang diautentikasi memiliki hak dan *privilege* yang diperlukan untuk mengakses dan melakukan tindakan pada sumber daya yang diminta.

Pada aktivitas mengidentifikasi subjek dan objek, subjek adalah entitas yang meminta akses, sedangkan objek adalah sumber daya yang akan ditindaklanjuti oleh subjek. Subjek dapat berupa pengguna manusia atau proses *software*.



Tindakan pada objek perlu ditangkap secara eksplisit, umumnya adalah operasi data *Create, Read, Update*, atau *Delete* (CRUD). Model kontrol akses yang dapat diterapkan, antara lain:

1) *Discretionary Access Control* (DAC)

Model kontrol akses ini membatasi akses ke objek berdasarkan identitas subjek dan/atau grup tempat objek tersebut berada. DAC diimplementasikan baik dengan menggunakan identitas atau peran. DAC sering diamati diimplementasikan dengan menggunakan *access control list* (ACL), hubungan antara individu (subjek) dan sumber daya (objek) bersifat langsung dan pemetaan individu ke sumber daya oleh pemiliknya.

2) *Non-Discretionary Access Control* (NDAC)

Model kontrol akses ini ditandai dengan *software* yang menerapkan kebijakan keamanan. Itu tidak bergantung pada kepatuhan subjek dengan kebijakan keamanan. Aspek *nondiscretionary* adalah bahwa itu tidak dapat dihindari dapat dikenakan pada semua subjek.

3) *Mandatory Access Control* (MAC)

Pada model kontrol akses ini, akses ke objek dibatasi untuk subjek berdasarkan sensitivitas informasi yang terkandung dalam objek. Sensitivitas diwakili oleh label. Hanya subjek yang memiliki *privilege* dan otorisasi formal yang sesuai yang diberi akses ke objek.

4) *Role Based Access Control* (RBAC)

Pada model kontrol akses ini, individu (subjek) memiliki akses ke sumber daya (objek) berdasarkan peran yang ditugaskan kepadanya. Izin untuk mengoperasikan objek seperti *Create, Read, Update*, atau *Delete* juga ditentukan dan ditentukan berdasarkan tanggung jawab dan wewenang dalam fungsi pekerjaan.

5) *Resource Based Access Control*

Pada model kontrol akses ini, akses diberikan berdasarkan sumber daya. Model kontrol akses berbasis sumber daya berguna dalam arsitektur yang didistribusikan dan multi-tier, termasuk arsitektur berorientasi layanan.

**f. Persyaratan Akuntabilitas**

Persyaratan akuntabilitas bertujuan untuk membangun catatan riwayat tindakan pengguna. Jejak audit dapat membantu mendeteksi saat pengguna yang tidak sah melakukan perubahan, atau pengguna yang berwenang membuat perubahan yang tidak sah, dimana keduanya merupakan kasus pelanggaran integritas.

## 2.2.2 Identifikasi Persyaratan Umum

**a. Persyaratan Manajemen Sesi**

Setelah autentikasi berhasil, *session identifier* (ID) dikeluarkan untuk pengguna dan *session ID* tersebut digunakan untuk melacak perilaku pengguna dan mempertahankan status terautentikasi untuk pengguna tersebut hingga sesi tersebut ditinggalkan atau status berubah dari terautentikasi menjadi tidak terautentikasi.

**b. Persyaratan Manajemen *Error* dan *Exception***

*Error* dan *exception* merupakan sumber potensial pengungkapan informasi.



### c. Persyaratan Manajemen Parameter Konfigurasi

Parameter dan *code* konfigurasi yang menyusun *software* membutuhkan perlindungan terhadap peretas. Parameter dan kode tersebut biasanya perlu diinisialisasi sebelum *software* dapat dijalankan.

## 2.2.3 Identifikasi Persyaratan Operasional

### a. Persyaratan Lingkungan *Deployment*

Persyaratan terkait tentang lingkungan dimana *software* akan di-*deploy* harus diidentifikasi dan dianalisis.

### b. Persyaratan Pengarsipan

Persyaratan pengarsipan harus secara eksplisit diidentifikasi karena arsip yang disimpan baik merupakan sarana untuk kelangsungan bisnis atau sebagai kebutuhan untuk memenuhi persyaratan peraturan atau kebijakan organisasi.

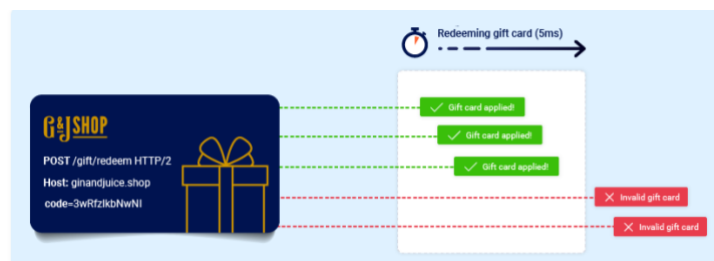
### c. Persyaratan Anti-Pembajakan (*Anti-Piracy*)

*Code obfuscation*, *code signing*, *anti-tampering*, lisensi, dan mekanisme perlindungan IP harus dimasukkan sebagai bagian dari dokumentasi persyaratan, terutama jika dalam bisnis pembuatan dan penjualan *software* komersial.

## 2.2.4 Identifikasi Persyaratan Lain

### a. Persyaratan Urutan dan Waktu

Cacat desain pada urutan dan waktu dalam *software* dapat menyebabkan apa yang umumnya dikenal sebagai *race condition* atau serangan *Time of Check/Time of Use* (TOC/TOU). Faktanya, *race condition* adalah salah satu kelemahan paling umum yang diamati dalam desain *software*.



Contoh *race condition*.

### b. Persyaratan Internasional

Persyaratan internasional dapat terdiri dari 2 (dua) jenis, yaitu hukum dan teknologi. Persyaratan hukum bertujuan agar tidak melanggar peraturan apa pun, sedangkan persyaratan teknologi bertujuan untuk menentukan karakter pengkodean dan mengarahkan tampilan.

### c. Persyaratan Pengadaan

Identifikasi dan penentuan persyaratan keamanan *software* tidak kalah pentingnya ketika diambil keputusan untuk membeli *software* daripada membangunnya sendiri. Pengendalian keamanan yang diperlukan untuk pengadaan *software*, antara lain:

- Identifikasi kebutuhan: organisasi harus mengidentifikasi kebutuhan sebelum melakukan pengadaan *software*.
- Spesifikasi pengadaan: spesifikasi pengadaan harus memuat klausul spesifik mengenai persyaratan keamanan, sertifikasi keamanan produk, ketersediaan

*source code*, persyaratan pembuangan data, preferensi terhadap teknologi lokal, serta persyaratan kompetensi dan keahlian tim pengembangan.

- Manajemen vendor: mencakup pengelolaan vendor yang menyediakan *hardware* dan *software*, layanan konsultasi, dan layanan terkelola (*managed services*).
- Jejak sumber daya: mengacu pada riwayat lengkap pergerakan aset dari asal ke organisasi. Proses akuisisi harus memastikan catatan lengkap mengenai jejak sumber daya di seluruh siklus hidup sistem. Jejak sumber daya harus mencakup rantai pasokan *hardware* dan *software* yang lengkap.
- *System life cycle*: keamanan harus dimasukkan ke dalam semua fase *system life cycle*, termasuk konseptualisasi *software*, pengumpulan persyaratan, desain, implementasi, pengujian, penerimaan, penerapan, pemeliharaan, dan pembuangan.
- Proses *commissioning* atau pengujian: proses *commissioning* dengan peran administrator dan melakukan penilaian postur keamanan sebelum *commissioning* sistem dan secara berkala selama implementasi dan ketika ada perubahan pada lingkungan.
- Proses penonaktifan: melakukan uji pencadangan dan pemulihan, serta migrasi data sebelum penonaktifan. Manajemen perubahan harus dilakukan untuk menginformasikan pihak-pihak terkait mengenai *decommissioning* sistem.
- Pembuangan: pembuangan yang aman dapat berupa penghancuran fisik dan/atau sanitasi data.

## 2.2 Klasifikasi Data

Tahap ini bertujuan untuk memastikan data atau informasi sebagai aset digital yang paling berharga akan terlindungi.

### 2.2.1 Tipe Data

Tahap ini bertujuan untuk mengklasifikasikan data ke dalam tipe-tipe sebagai berikut:

#### a. Data Terstruktur

Data terstruktur mengacu pada data yang diorganisasikan ke dalam struktur yang dapat diidentifikasi. Contohnya data terstruktur dalam *database* (disimpan dalam bentuk kolom dan baris).

#### b. Data Tidak Terstruktur

Data tidak terstruktur merujuk pada data yang tidak memiliki struktur yang dapat diidentifikasi. Contohnya data tidak terstruktur termasuk gambar, video, *email*, dokumen, dan teks.

### 2.2.2 Memberi Label pada Data

Pelabelan data adalah upaya untuk memberikan label sesuai klasifikasi data pada data, berdasarkan dampak potensial terhadap aspek kerahasiaan, integritas, dan ketersediaan (C-I-A).

### 2.2.3 Kepemilikan dan Peran pada Data

Keputusan untuk mengklasifikasikan data, siapa yang memiliki hak akses, apa tingkat akses yang dimiliki, dan keputusan lainnya adalah keputusan yang harus dibuat oleh pemilik data.

#### 2.2.4 Data Lifecycle Management (DLM)

Pendekatan berbasis kebijakan yang melibatkan prosedur dan praktik untuk melindungi data di sepanjang *information life cycle*, dimulai sejak data dibuat hingga data dihapus.

#### 2.2.5 Persyaratan Privasi

Klasifikasi data dapat membantu dalam mengidentifikasi data yang perlu menerapkan persyaratan perlindungan privasi. Mengkategorikan data ke dalam berbagai tingkatan privasi, berdasarkan dampak pada pengungkapan, pengubahan, dan/atau penghancuran, dapat memberikan wawasan untuk memastikan bahwa tingkat kontrol privasi yang sesuai telah diterapkan.

Pedoman *best practice* untuk privasi data yang perlu disertakan dalam analisis, desain, dan arsitektur persyaratan *software* dapat diatasi jika mematuhi aturan berikut:

- Jika data tidak dibutuhkan, maka jangan mengumpulkannya.
- Jika data perlu dikumpulkan hanya untuk diproses, maka kumpulkan setelah memberi tahu pengguna bahwa organisasi mengumpulkan informasi pengguna dan pengguna telah menyetujuinya, tetapi jangan menyimpannya.
- Jika data perlu dikumpulkan untuk pemrosesan dan penyimpanan, maka kumpulkan dengan persetujuan pengguna, dan simpan hanya untuk periode penyimpanan eksplisit yang sesuai dengan kebijakan organisasi dan/atau persyaratan peraturan.
- Jika terdapat kebutuhan untuk mengumpulkan dan menyimpan data, maka jangan mengarsipkannya jika data tersebut telah habis masa pakainya dan tidak ada persyaratan retensi.

Persyaratan dan kontrol privasi adalah sebagai berikut:

##### a. Anonimisasi Data

Dengan menghapus pengidentifikasi pribadi dari data secara permanen dan menyeluruh, anonimitas dapat terjamin. Anonimisasi adalah proses menghapus informasi pribadi dari data. Data anonim tidak dapat ditautkan ke satu akun individual mana pun.

Teknik anonimisasi berguna untuk memastikan privasi data, antara lain: penggantian (*substitution*), penghilangan (*suppression*), data pribadi diganti menggunakan bentuk yang lebih umum (*generalization*), dan pengacakan yang melibatkan perubahan acak pada data (*randomization*).

Definition	Personal sensitive data	Pseudonymous data	Anonymous data
	This is the full data including personal and special data.	IDs are replaced with pseudonyms. Sensitive data is encrypted.	IDs removed & sensitive data randomised/generalised.
	Name: John Briggs Date of birth: 14.04.87 Email: jb89@mail.com User ID: john_briggs_89 Health: type 1 diabetes	Names: User-78463 Date of birth: 14.04.87 Email: [REDACTED] User ID: [REDACTED] Health: type 1 diabetes	Sex: Male Age: 30-49 Health: type 1 diabetes
	* special data includes health, gender, genetics, biometrics, ethnic origin, sexuality, politics & religion		

Contoh anonimisasi data.

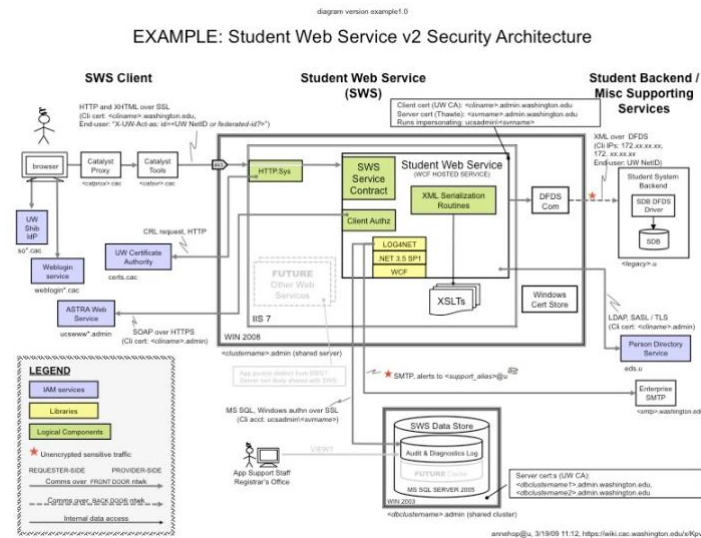
##### b. Penghapusan Data

Semua *software* bersifat rentan, kecuali jika *software* dan data yang diproses, ditransmisikan, disimpan, dan dihapus dengan cara yang aman. Hal ini sangat penting untuk diperhatikan, jika datanya sensitif atau terdapat data pribadi di dalamnya.

Sebagian besar peraturan privasi mensyaratkan penerapan kebijakan dan prosedur untuk mengatur penghapusan data pribadi dan sanitasi *hardwaredan* media penyimpanan elektronik sebelum dikondisikan untuk dapat digunakan kembali.

### c. Security Model

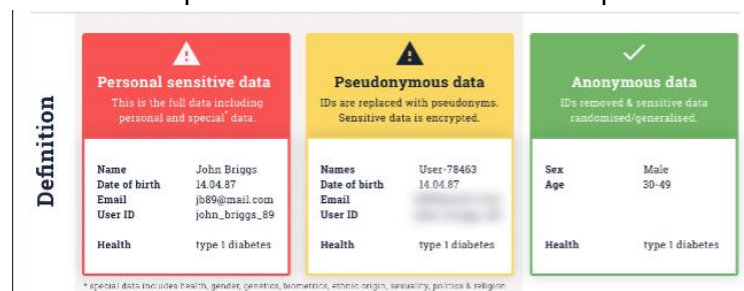
*Security model* adalah abstraksi formal dari kebijakan keamanan yang terdiri dari sekumpulan persyaratan keamanan yang perlu menjadi bagian dari *software*, sehingga *software* tahan terhadap serangan, dapat mentolerir serangan yang tidak dapat dilawan, dan dapat pulih dengan cepat dari keadaan yang tidak diinginkan jika dikompromikan.



Contoh security model.

### d. Pseudonimisasi (*Pseudonymization*)

Pseudonimisasi data adalah manajemen data dan prosedur *de-identification* dimana bidang informasi yang dapat diidentifikasi secara pribadi dalam data rekaman diganti dengan 1 (satu) atau lebih pengidentifikasi buatan atau samaran. Sebuah samaran tunggal untuk setiap bidang yang diganti atau kumpulan bidang yang diganti menyebabkan data rekaman kurang dapat diidentifikasi namun tetap cocok untuk analisis data dan pemrosesan data.



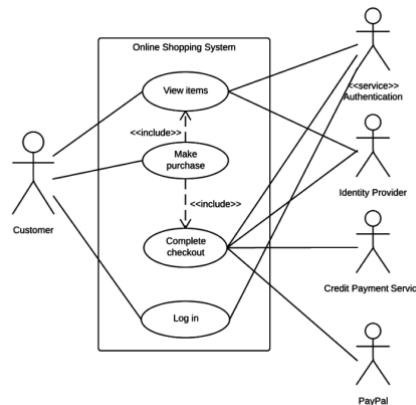
Contoh pseudonimisasi data.

## 2.3 Pemodelan Use Case dan Misuse Case

Tahap ini bertujuan untuk mengidentifikasi kemungkinan perilaku buruk dan merekomendasikan persyaratan keamanan yang relevan berdasarkan fungsionalitas untuk *software* yang dikembangkan.

### 2.3.1 Menganalisis Skenario *Use Case*

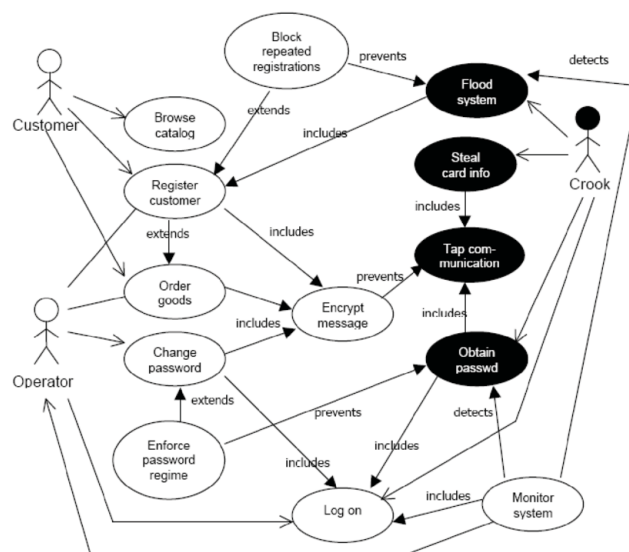
*Use case software* menggambarkan perilaku yang diinginkan oleh pemilik *software*, sehingga akan mengidentifikasi perilaku *software*. Pemodelan dan pembuatan diagram *use case* sangat berguna untuk menentukan persyaratan ini. Pemodelan ini efektif dalam mengurangi persyaratan bisnis yang ambigu dan tidak lengkap dengan secara eksplisit menentukan dengan tepat kapan dan dalam kondisi apa perilaku tertentu terjadi. Pemodelan *use case* meliputi pengidentifikasian aktor (individu, peran, atau non-manusia), perilaku *software* yang dimaksudkan (*use case*), dan urutan serta hubungan antara aktor dan *use case*.



Contoh diagram *use case*.

### 2.3.2 Menganalisis Skenario *Misuse Case*

*Misuse case* memodelkan skenario negatif untuk mengidentifikasi kelemahan yang menjadi ancaman. Skenario negatif adalah perilaku *software* yang tidak diinginkan oleh pemilik *software* dalam konteks *use case*. *Misuse case* memberikan wawasan tentang ancaman yang dapat terjadi terhadap *software*. Dalam pemodelan *misuse case*, akan dibuat pemodelan untuk pelaku yang melakukan kesalahan, dan skenario atau perilaku yang tidak diinginkan. *Misuse case* dapat bersifat disengaja atau tidak disengaja. *Misuse case* dapat dibuat melalui *brainstorming* skenario negatif seperti penyerang.



Contoh diagram *misuse case*.

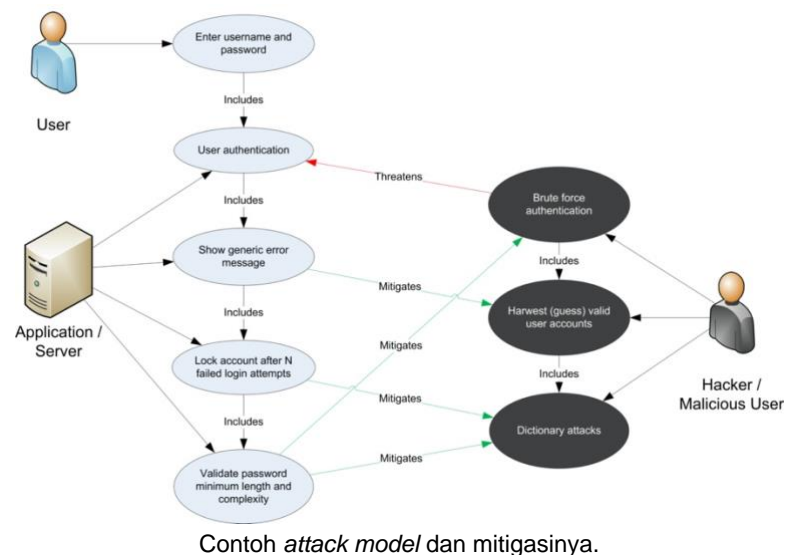
### 2.3.3 Membuat *Attack Model*

Untuk membuat *attack model* dengan pertimbangan eksplisit dari serangan yang diketahui, diperlukan serangkaian persyaratan dan daftar ancaman, kemudian dapat ditelusuri daftar serangan yang diketahui satu per satu dan memikirkan apakah terdapat serangan yang sama berlaku untuk *software*. Untuk membuat *attack model*, lakukan langkah berikut:

- Pilih pola serangan yang relevan dengan *software*. Buatlah *misuse case* di sekitar pola serangan tersebut.
- Sertakan siapa saja yang dapat memperoleh akses ke *software* karena sumber ancaman harus mencakup semua potensi sumber bahaya terhadap *software*.

### 2.3.4 Memilih Kontrol Mitigasi

Untuk mengusulkan kontrol mitigasi berdasarkan serangan yang diidentifikasi dari langkah sebelumnya, soroti intinya dengan mengusulkan kontrol keamanan. Hal ini membantu dalam memenuhi persyaratan keamanan.



Contoh *attack model* dan mitigasinya.

## 2.4 Manajemen Risiko

Manajemen risiko pada pengembangan *software* bertujuan untuk:

- Mengamankan *software* yang menyimpan, memproses, atau mengirimkan informasi organisasi.
- Memungkinkan pihak manajemen membuat keputusan manajemen risiko yang terinformasi dengan baik untuk mendukung sumber daya yang dibutuhkan dalam pengembangan *software*.
- Membantu pihak manajemen dalam mengotorisasi (atau mengakreditasi) *software* berdasarkan dokumentasi pendukung yang dihasilkan dari kinerja manajemen risiko.

### 2.4.1 Penilaian Risiko

Penilaian risiko pada pengembangan *software* bertujuan untuk:

- Menentukan sejauh mana potensi ancaman dan risiko yang terkait dengan *software*.
- Untuk mengidentifikasi kontrol keamanan yang tepat untuk mengurangi atau menghilangkan risiko selama proses mitigasi risiko.

Metodologi penilaian risiko mencakup 9 (sembilan) langkah utama, dimana langkah 2, 3, 4, dan 6 dapat dilakukan secara paralel setelah langkah 1 selesai. Langkah-langkahnya seperti di bawah ini:



**a. Langkah 1: Karakterisasi Sistem**

Karakterisasi sistem bertujuan untuk menentukan ruang lingkup penilaian risiko. Pada langkah ini, batas-batas *software* diidentifikasi, bersama dengan sumber daya dan informasi yang membentuk sistem. Karakterisasi *software* menetapkan ruang lingkup upaya penilaian risiko, menggambarkan batasan otorisasi operasional dan memberikan informasi yang penting untuk menentukan risiko, misalnya, *hardware*, *software*, konektivitas sistem, dan unit kerja yang bertanggung jawab atau personel pendukung.

**b. Langkah 2: Identifikasi Ancaman**

Identifikasi ancaman bertujuan untuk mengidentifikasi sumber ancaman potensial dan menyusun pernyataan ancaman yang mencantumkan sumber ancaman potensial yang berlaku untuk *software* yang sedang dievaluasi. Sumber ancaman didefinisikan sebagai apa saja keadaan atau peristiwa yang berpotensi menyebabkan kerusakan pada *software*. Sumber ancaman secara umum dapat berupa alam, manusia, atau lingkungan.

**c. Langkah 3: Identifikasi Kerentanan**

Analisis ancaman terhadap *software* harus mencakup analisis kerentanan yang terkait dengan lingkungan *software*. Identifikasi kerentanan bertujuan untuk mengembangkan daftar kerentanan *software* (cacat atau kerentanan) yang dapat dimanfaatkan oleh sumber ancaman potensial.

**d. Langkah 4: Analisis Kontrol**

Analisis kontrol bertujuan untuk:

- 1) Menganalisis kontrol yang telah diterapkan atau direncanakan akan diterapkan untuk meminimalkan atau menghilangkan kemungkinan (atau probabilitas) dari sumber ancaman yang mengeksploitasi kerentanan *software*.
- 2) Memperoleh tingkat kemungkinan yang menunjukkan kemungkinan terjadinya kerentanan potensial yang dapat dieksekusi pada konstruksi lingkungan ancaman terkait (langkah 5), maka penerapan dari kontrol saat ini atau kontrol yang direncanakan harus dipertimbangkan.

**e. Langkah 5: Penentuan Tingkat Kemungkinan**

Untuk memperoleh tingkat kemungkinan yang menunjukkan kemungkinan terjadinya kerentanan potensial yang dapat dieksploitasi pada konstruksi lingkungan ancaman terkait, faktor-faktor yang mengatur berikut ini harus dipertimbangkan:

- 1) Motivasi dan kapabilitas sumber ancaman.
- 2) Sifat kerentanan.
- 3) Keberadaan dan efektivitas kontrol saat ini.

**f. Langkah 6: Penentuan Tingkat Dampak**

Penentuan tingkat dampak bertujuan untuk menentukan dampak buruk yang dihasilkan dari keberhasilan eksploitasi sumber ancaman terhadap kerentanan. Sebelum memulai analisis dampak, perlu untuk mendapatkan informasi yang diperlukan berikut ini:

- 1) Misi sistem (misalnya, proses yang dilakukan oleh *software*).
- 2) Kekritisan sistem dan data (misalnya, nilai atau kepentingan sistem bagi organisasi).
- 3) Sensitivitas sistem dan data.

**g. Langkah 7: Penentuan Tingkat Risiko**

Penentuan tingkat risiko bertujuan untuk menilai tingkat risiko untuk *software*. Penentuan tingkat risiko didapatkan dari kombinasi antara tingkat kemungkinan (langkah 5) dan tingkat dampak (langkah 6).

**h. Langkah 8: Rekomendasi Kontrol**

Kontrol keamanan harus disediakan untuk dapat mengurangi atau menghilangkan risiko yang teridentifikasi sesuai dengan operasional organisasi ke tingkat risiko yang dapat diterima. Faktor-faktor berikut harus dipertimbangkan dalam merekomendasikan kontrol dan solusi alternatif untuk meminimalkan atau menghilangkan risiko yang teridentifikasi:

- 1) Efektivitas opsi yang direkomendasikan.
- 2) Aspek regulasi.
- 3) Kebijakan organisasi.
- 4) Dampak operasional.
- 5) Keamanan dan keandalan.

**i. Langkah 9: Dokumentasi Hasil**

Setelah penilaian risiko selesai, maka hasilnya harus didokumentasikan dalam laporan. Laporan penilaian risiko adalah laporan yang membantu pihak manajemen dan pemilik risiko dalam membuat keputusan tentang kebijakan, prosedur, anggaran, dan operasional sistem dan manajemen perubahan. Laporan penilaian risiko harus disajikan pendekatan sistematis dan analitis untuk menilai risiko sehingga pihak manajemen dapat memahami risiko dan mengalokasikan sumber daya untuk mengurangi dan memperbaiki kerugian potensial.

## **2.4.2 Mitigasi Risiko**

Tahap ini bertujuan untuk memprioritaskan, mengevaluasi, dan menerapkan kontrol pengurangan risiko yang sesuai yang direkomendasikan dari proses penilaian risiko.

**a. Opsi Mitigasi Risiko**

Organisasi harus mempertimbangkan untuk memilih dari beberapa opsi pada mitigasi risiko. Mungkin tidak praktis untuk mengatasi semua risiko yang teridentifikasi, jadi prioritas harus diberikan pada pasangan ancaman dan kerentanan yang berpotensi menyebabkan dampak atau kerugian yang signifikan. Pilihan mitigasi risiko adalah:

- 1) Menerima risiko, yaitu menerima potensi risiko dengan terus mengoperasikan *software* atau menerapkan kontrol untuk menurunkan risiko ke tingkat yang dapat diterima.
- 2) Menghindari risiko, yaitu menghindari risiko dengan menghilangkan penyebab dan/atau konsekuensi risiko. Misalnya, melewati fungsi tertentu dari *software* atau mematikan *software* saat risiko teridentifikasi.
- 3) Mitigasi risiko, yaitu membatasi risiko dengan menerapkan kontrol yang meminimalkan dampak merugikan dari ancaman yang menjalankan kerentanan. Misalnya penggunaan pendukung, pencegahan, kontrol yang bersifat detektif.
- 4) Mentransfer risiko, yaitu mentransfer risiko dengan menggunakan opsi lain untuk mengkompensasi kerugian, seperti membeli asuransi.



- 5) Perencanaan risiko, yaitu mengelola risiko dengan mengembangkan rencana mitigasi risiko yang memprioritaskan, mengimplementasikan, dan memelihara kontrol.
- 6) Penelitian, yaitu menurunkan risiko kerugian dengan mengakui adanya kerentanan dan meneliti kontrol keamanan untuk memperbaiki kerentanan.

**b. Strategi Mitigasi Risiko**

Tindakan untuk mengurangi risiko dari ancaman manusia yang disengaja, antara lain:

- 1) Ketika terdapat kerentanan, maka harus menerapkan perbaikan untuk mengurangi tingkat kemungkinan pada kerentanan yang dieksploitasi.
- 2) Ketika kerentanan dapat dieksploitasi, maka terapkan perlindungan berlapis, desain arsitektural, dan kontrol administratif untuk meminimalkan risiko atau mencegah insiden.
- 3) Ketika biaya untuk mengatasi penyerang kurang dari keuntungan potensial, maka terapkan perlindungan untuk mengurangi motivasi penyerang dengan meningkatkan biaya untuk mengatasi penyerang. Misalnya, penggunaan kontrol *software* untuk membatasi apa yang dapat diakses dan dilakukan oleh pengguna *software* yang secara signifikan dapat mengurangi faktor-faktor yang menguntungkan penyerang.
- 4) Ketika kerugian terlalu besar, maka terapkan prinsip-prinsip desain, desain arsitektural, dan perlindungan teknis dan non-teknis untuk membatasi tingkat serangan, sehingga dapat mengurangi potensi kerugian.

**c. Pendekatan pada Implementasi Kontrol Keamanan**

Langkah-langkah implementasi kontrol keamanan, antara lain:

- 1) Langkah 1: Prioritaskan tindakan  
Memprioritaskan tindakan implementasi kontrol keamanan berdasarkan tingkat risiko.
- 2) Langkah 2: Mengevaluasi opsi kontrol keamanan  
Langkah ini bertujuan untuk memilih opsi kontrol yang paling tepat dalam meminimalkan risiko. Menganalisis kelayakan (misalnya, kompatibilitas, penerimaan pengguna) dan efektivitas (misalnya, tingkat perlindungan, dan tingkat mitigasi risiko) dari opsi kontrol keamanan yang direkomendasikan.
- 3) Langkah 3: Lakukan analisis biaya dan manfaat  
Langkah ini bertujuan untuk membantu pihak manajemen dalam pengambilan keputusan dan mengidentifikasi kontrol keamanan yang hemat biaya, maka harus dilakukan analisis biaya dan manfaat.
- 4) Langkah 4: Pemilihan kontrol keamanan  
Langkah ini bertujuan untuk menentukan kontrol keamanan yang paling hemat biaya untuk mengurangi risiko terhadap misi organisasi.
- 5) Langkah 5: Menetapkan tanggung jawab  
Langkah ini bertujuan untuk mengidentifikasi dan menugaskan orang yang tepat (staf internal atau staf vendor eksternal) yang memiliki keahlian dan keterampilan yang sesuai untuk menerapkan kontrol yang dipilih.
- 6) Langkah 6: Membuat rencana implementasi kontrol keamanan  
Langkah ini bertujuan untuk membuat rencana implementasi kontrol keamanan (atau rencana aksi).
- 7) Langkah 7: Menerapkan kontrol keamanan terpilih

Langkah ini bertujuan untuk menerapkan kontrol yang dapat menurunkan tingkat risiko tetapi tidak menghilangkan risiko (risiko residual).

**d. Kategori Kontrol Keamanan**

Pengkategorian kontrol keamanan bertujuan untuk mempertimbangkan kontrol keamanan teknis, manajemen, dan operasional, atau kombinasi dari kontrol tersebut, serta untuk memaksimalkan efektivitas kontrol untuk *software* dan organisasi.

- 1) Kontrol keamanan teknis untuk mitigasi risiko dapat dikonfigurasi untuk melindungi dari jenis ancaman tertentu. Kontrol ini dapat berkisar dari tindakan sederhana hingga kompleks dan biasanya melibatkan arsitektur *software*, ilmu teknik, dan sistem keamanan yang merupakan campuran dari *hardware*, *software*, dan *firmware*.
- 2) Kontrol keamanan manajemen, bersama dengan kontrol teknis dan operasional, diterapkan untuk mengelola dan mengurangi risiko kerugian dan untuk melindungi misi organisasi. Kontrol keamanan manajemen fokus pada penetapan kebijakan, pedoman, dan standar perlindungan informasi, yang dilakukan melalui prosedur operasional untuk memenuhi tujuan dan misi organisasi.

**e. Analisis Biaya dan Manfaat**

Analisis biaya dan manfaat bertujuan untuk mengalokasikan sumber daya dan menerapkan kontrol yang hemat biaya pada organisasi, setelah mengidentifikasi semua kemungkinan kontrol dan mengevaluasi kelayakan dan keefektifannya. Analisis biaya dan manfaat harus dilakukan untuk setiap kontrol yang diusulkan untuk menentukan kontrol mana yang diperlukan dan sesuai untuk keadaan organisasi.

**f. Risiko Residual**

Analisis risiko residual bertujuan untuk mengetahui sejauh mana pengurangan risiko yang dihasilkan oleh kontrol baru atau kontrol yang ditingkatkan dalam hal pengurangan tingkat kemungkinan atau tingkat dampak dari sumber ancaman.

**2.4.3 Evaluasi dan Penilaian Risiko**

Tahap ini bertujuan untuk menekankan praktik manajemen risiko yang baik dan perlunya evaluasi dan penilaian risiko yang berkelanjutan. Proses evaluasi dan penilaian harus dilakukan secara berkala, sehingga perlu ada jadwal khusus. Namun demikian, juga harus cukup fleksibel untuk memungkinkan perubahan jika diperlukan, seperti perubahan besar pada *software* dan lingkungan pemrosesan karena perubahan yang dihasilkan dari kebijakan dan teknologi baru.

## Bab III - Desain Keamanan (*Security Design*)

Desain keamanan bertujuan untuk:

- Merancang arsitektur yang aman dengan mempertimbangkan elemen keamanan.
- Mengamankan arsitektur *software* dengan memahami dan menganalisis ancaman terhadap *software* sebelum dibangun.

### 3.1 Desain Keamanan Inti (*Core Security Design*)

Desain keamanan inti bertujuan untuk merancang *software* untuk mengatasi elemen *core security* dari aspek kerahasiaan, integritas, ketersediaan, autentikasi, otorisasi, dan audit.

#### 3.1.1 Desain Kerahasiaan

Desain kerahasiaan bertujuan untuk merancang pelindungan terhadap pengungkapan informasi dapat dicapai, yaitu menggunakan teknik kriptografi dan penyamaran (*masking*). Penyamaran (*masking*) berguna untuk memberi pelindungan dari pengungkapan informasi saat data ditampilkan di layar atau dicetak. Sedangkan kriptografi digunakan untuk memberikan jaminan kerahasiaan pada saat data ditransmisikan atau disimpan di tempat penyimpanan data transaksional atau diarsipkan secara *offline*.

Faktor yang mempengaruhi kekuatan pelindungan kriptografi antara lain:

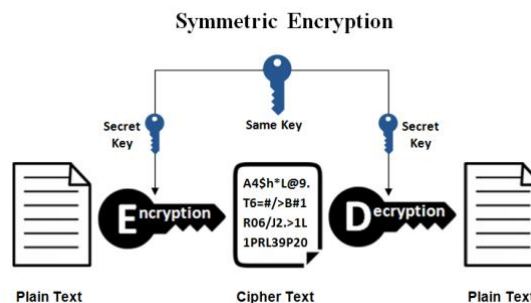
- Algoritma enkripsi, yaitu proses pengubahan informasi dari teks yang terbaca (*plaintext*) menjadi teks yang tidak terbaca (*ciphertext*).
- Ukuran kunci, yaitu panjang kunci yang digunakan dalam algoritma.
- Manajemen kunci, yaitu pengelolaan kunci enkripsi yang meliputi pembangkitan, pertukaran, penyimpanan, rotasi, pengarsipan, dan pemusnahan.

Time to brute force password space, assuming 10,000 attempts per second			
	Lowercase (26 letters)	Uppercase, lowercase, digits (62 characters)	Uppercase, lowercase, digits, punctuation (94 characters)
Length = 5 characters	19 minutes	1 day	8 days
Length = 6 characters	8 hours	65 days	2 years
Length = 7 characters	9 days	11 years	200 years
Length = 8 characters	241 days	692 years	19,000 years
Length = 9 characters	17 years	42,000 years	1.8 million years

Contoh kekuatan kriptografi dari algoritma AES.

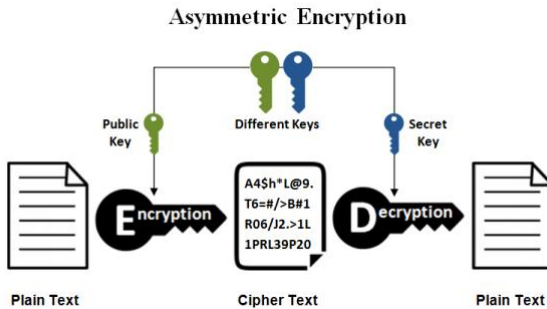
Algoritma enkripsi pada kriptografi terutama terdiri dari 2 (dua) jenis:

- Algoritma simetrik, yaitu algoritma enkripsi yang menggunakan 1 (satu) kunci untuk operasi enkripsi dan dekripsi yang digunakan bersama antara pengirim dan penerima.



Ilustrasi algoritma simetrik.

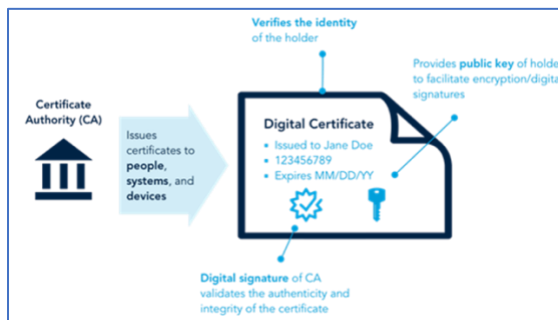
- b. Algoritma asimetrik, yaitu algoritma enkripsi yang menggunakan 2 (dua) kunci, yaitu kunci yang dirahasiakan disebut sebagai kunci *private*, dan kunci yang diungkapkan kepada siapa saja yang perlu melakukan komunikasi aman dan ditampilkan secara publik disebut sebagai kunci *public*.



Ilustrasi algoritma asimetrik.

Algoritma asimetrik digunakan pada:

- 1) Sertifikat elektronik, yaitu dokumen elektronik yang menentukan format untuk kunci *public* dan digunakan oleh siapa saja untuk memverifikasi keaslian sertifikat digital itu sendiri karena di dalamnya terdapat sertifikat digital dari *Certificate Authority* (CA).



Ilustrasi sertifikat elektronik.

- 2) Tanda tangan elektronik, yaitu skema matematis yang digunakan untuk memberikan verifikasi identitas dan memastikan bahwa data atau pesan tidak dirusak karena tanda tangan elektronik yang digunakan untuk menandatangani pesan tidak dapat dengan mudah ditiru kecuali jika dikompromikan. Tanda tangan elektronik juga menyediakan bukti untuk nir-penyangkalan.

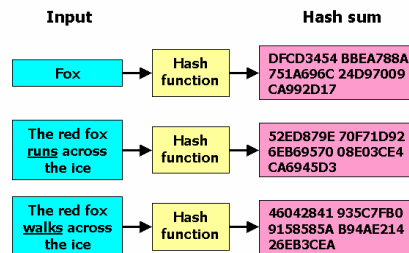


Ilustrasi tanda tangan elektronik.

### 3.1.2 Desain Integritas

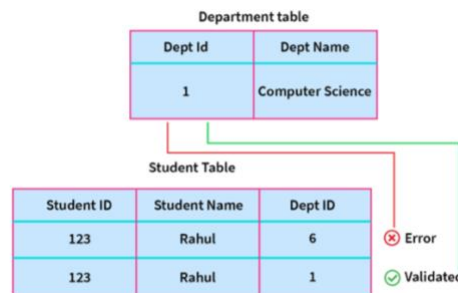
Desain integritas bertujuan untuk memastikan bahwa tidak ada modifikasi *software* atau data yang tidak sah dengan menggunakan salah satu dari teknik berikut atau kombinasi dari teknik-teknik berikut:

- Hashing* (fungsi *hash*), yaitu fungsi yang digunakan untuk memadatkan *input* dengan panjang variabel menjadi *output* berukuran tetap yang tidak dapat diubah yang dikenal sebagai intisari (*digest*) pesan atau nilai *hash*.



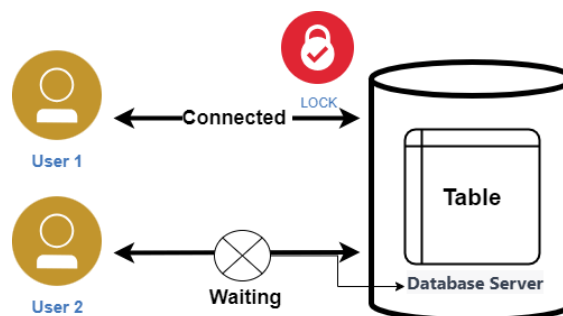
Ilustrasi fungsi *hash*.

- Integritas referensi (*referential integrity*), yaitu jaminan integritas data, terutama dalam *relational database management system* (RDBMS) yang memastikan bahwa data tidak dibiarkan dalam keadaan tanpa relasi. Menggunakan kunci *primary* dan kunci *foreign* terkait dalam *database* untuk memastikan integritas data.



Ilustrasi integritas referensi.

- Penguncian sumber daya (*resource locking*), yaitu ketika 2 (dua) operasi bersamaan tidak diizinkan pada objek yang sama (misalnya pada *record* di *database*), karena salah satu operasi mengunci *record* itu agar tidak mengizinkan perubahan apa pun hingga selesai operasinya.

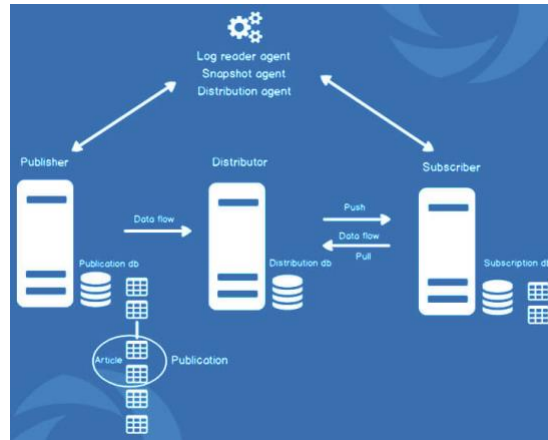


Ilustrasi *database lock*.

### 3.1.3 Desain Ketersediaan

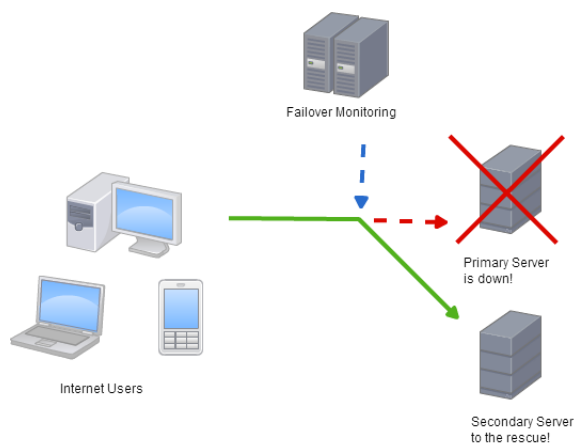
Desain ketersediaan bertujuan untuk mendapatkan perlindungan terhadap upaya penghancuran dan *Denial of Service* (DoS) dengan pembangunan *software* yang tepat. Teknik yang digunakan untuk merancang *software* untuk mendukung aspek ketersediaan antara lain:

- a. Replikasi (*replication*), yaitu penyediaan infrastruktur jika terjadi satu titik kegagalan (*single point of failure*) pada kondisi tidak adanya kemampuan redundansi. Namun jika kegagalan terjadi, maka akan terdapat gangguan operasional yang merugikan pengguna.



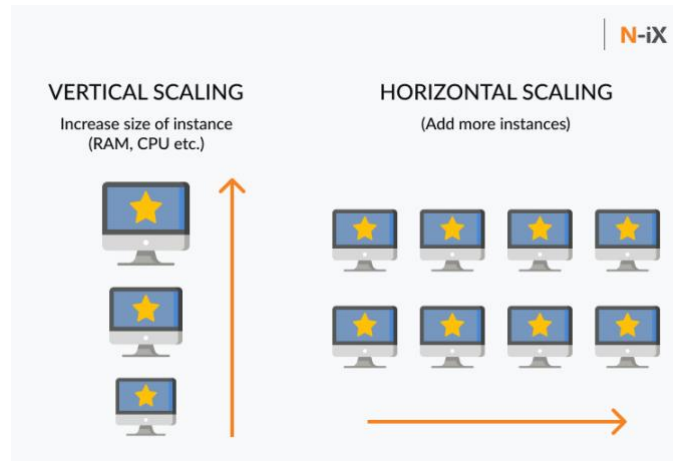
Ilustrasi infrastruktur replikasi pada *server database*.

- b. *Failover*, yaitu penyediaan infrastruktur dengan peralihan otomatis dari *software* transaksional yang aktif/*server*/sistem/komponen *hardware*/ jaringan ke sistem yang disiagakan (redundan). Namun, peralihan bersifat manual.



Ilustrasi infrastruktur *failover*.

- c. *Skalabilitas* (*scalability*), yaitu penyediaan infrastruktur pada peningkatan beban pekerjaan sistem tanpa penurunan fungsi atau kinerjanya.



Ilustrasi skalabilitas pada infrastruktur.

### 3.1.4 Desain Autentikasi

Desain autentikasi bertujuan untuk menentukan jenis autentikasi yang diperlukan seperti yang ditentukan dalam dokumentasi persyaratan. Direkomendasikan menggunakan autentikasi multi-faktor atau autentikasi dua faktor (2FA) untuk mengautentikasi entitas (pengguna atau sumber daya) yang memberikan keamanan yang lebih tinggi. Jika ada kebutuhan untuk menerapkan SSO, dimana identitas entitas yang dinyatakan diverifikasi sekali dan kredensial yang diverifikasi diteruskan ke sistem atau aplikasi lain, maka perlu menggunakan token. Sangat penting untuk mempertimbangkan desain *software* yang baik dampak kinerja maupun keamanannya.

### 3.1.5 Desain Otorisasi

Desain otorisasi bertujuan untuk memberikan perhatian pada dampak kinerja, dan prinsip pemisahan tugas dan hak istimewa terkecil (*least privilege*). Jenis otorisasi yang akan diterapkan sesuai dengan persyaratan harus ditentukan, seperti peran atau otorisasi berbasis sumber daya. Jika peran digunakan untuk otorisasi, maka harus dipastikan bahwa tidak ada peran yang bertentangan karena belum diterapkannya prinsip pemisahan tugas. Misalnya, pengguna tidak dapat berperan sebagai *teller* dan juga sebagai auditor untuk transaksi keuangan. Merancang untuk otorisasi dapat dilakukan dengan menggunakan manajemen hak, yaitu tentang kontrol akses granular (spesifik dan presisi untuk setiap pengguna).

Mekanisme kontrol akses yang sesuai untuk objek yang bersifat umum antara lain:

#### a. Direktori

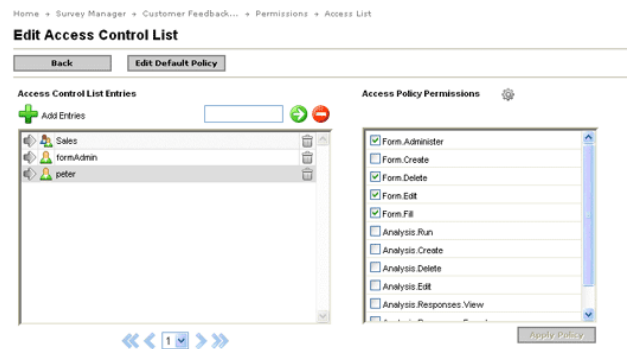
Salah satu cara sederhana untuk melindungi objek adalah dengan menggunakan mekanisme yang berfungsi seperti direktori *file*. Hal ini lebih mudah daripada melindungi *file* (kumpulan objek) dari pengguna sistem komputasi (kumpulan subjek) dimana setiap *file* memiliki pemilik unik yang memiliki hak akses (hak yang menyatakan siapa yang memiliki akses apa) dan hak untuk mencabut akses ke siapa pun kapan saja. Lebih mudah untuk mengelola hak akses pengguna pada direktori *file* dimana tercantum semua *file* yang dapat diakses oleh pengguna tersebut.



Ilustrasi akses kontrol berbasis direktori.

### b. Access Control List (ACL)

Ada 1 (satu) daftar untuk setiap objek, dan daftar tersebut menunjukkan semua subjek yang memiliki akses ke objek dan apa saja hak aksesnya. Pendekatan ini berbeda dari daftar direktori karena terdapat satu daftar kontrol akses per objek, sementara direktori dibuat untuk setiap subjek.



Ilustrasi Access Control List (ACL) pada software.

### c. Matriks Kontrol Akses (Access Control Matrix)

Matriks kontrol akses merupakan tabel dimana setiap baris mewakili subjek, setiap kolom mewakili objek, dan setiap entri adalah himpunan hak akses untuk subjek ke objek itu.

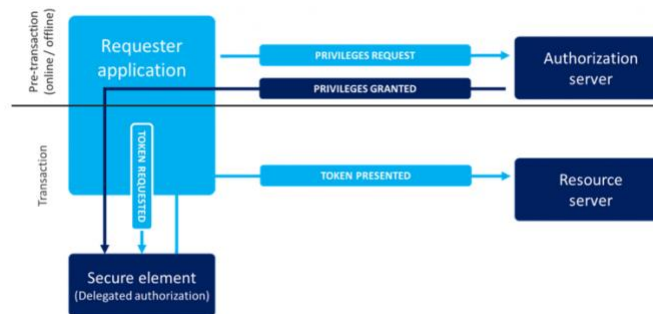
	File 1	File 2	File 3	...	File n
User 1	read	write		-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
...					
User m	read	write	read	write	read

Ilustrasi matriks kontrol akses.



#### d. Kapabilitas (*Capability*)

Kontrol akses berbasis kapabilitas dalam hal ini menggunakan token yang tidak dapat dipalsukan, yang memberikan hak tertentu atas suatu objek. Secara teori, subjek dapat membuat objek baru dan dapat menentukan operasi yang diperbolehkan pada objek tersebut. Misalnya, pengguna dapat membuat objek, seperti *file*, segmen data, atau subproses, dan dapat menentukan jenis operasi yang dapat diterima, seperti *read*, *write*, dan *execute*. Tetapi pengguna juga dapat membuat objek yang benar-benar baru, seperti struktur data baru, dan dapat menentukan jenis akses yang sebelumnya tidak diketahui oleh *software*.



Ilustrasi kontrol akses berbasis kapabilitas.

#### e. Kontrol Akses Berorientasi Prosedur (*Procedure Oriented Access Control*)

Merupakan keberadaan prosedur yang mengontrol akses ke objek (misalnya, dengan melakukan autentikasi penggunaannya sendiri untuk memperkuat perlindungan dasar yang disediakan oleh sistem operasi dasar). Prosedur membentuk kapsul di sekitar objek dan hanya mengizinkan akses tertentu yang ditentukan.

### 3.1.6 Desain Akuntabilitas

Desain akuntabilitas bertujuan untuk mengaudit *software* terutama jika terjadi pelanggaran, terutama untuk tujuan forensik digital. Data *log* harus mencakup aspek *who*, *what*, *where* dan *when* pada operasi di *software*. Sebagai bagian dari *who*, penting untuk tidak melupakan aktor non-manusia seperti proses *batch*, layanan, atau *daemon*.

## 3.2 Desain Tambahan

Tahap ini bertujuan untuk membahas pertimbangan desain lain yang diperlukan saat membangun *software* yang aman.

### 3.2.1 Bahasa Pemrograman

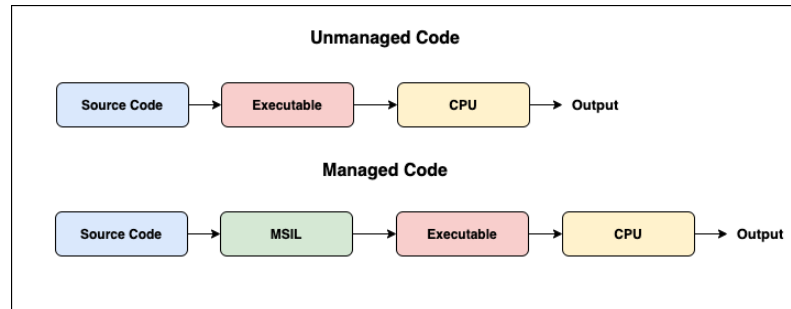
Bahasa pemrograman yang akan digunakan harus ditentukan pada tahap desain karena akan membawa risiko atau manfaat keamanan yang melekat. Ada dua (2) jenis utama bahasa pemrograman:

#### a. Kode Tidak Terkelola (*Unmanaged Code*)

Misalnya, C/C++. Pada bahasa pemrograman ini, eksekusi kode tidak dikelola oleh lingkungan eksekusi *runtime* mana pun tetapi langsung dieksekusi oleh sistem operasi.

**b. Kode terkelola (*Managed code*)**

Misalnya Java dan .NET, C#, VB.Net. Pada bahasa pemrograman ini, eksekusi kode tidak dilakukan oleh sistem operasi secara langsung, melainkan dikelola oleh lingkungan *runtime*. Layanan keamanan dan non-keamanan seperti manajemen memori, *exception handling*, *bound checking*, *garbage collection*, dan pemeriksaan keamanan yang diberikan oleh lingkungan *runtime*, dan pemeriksaan keamanan yang dilakukan sebelum kode dijalankan.



Perbandingan antara *unmanaged code* dan *managed code*.

**3.2.2 Jenis, Format, Jangkauan, dan Panjang Data**

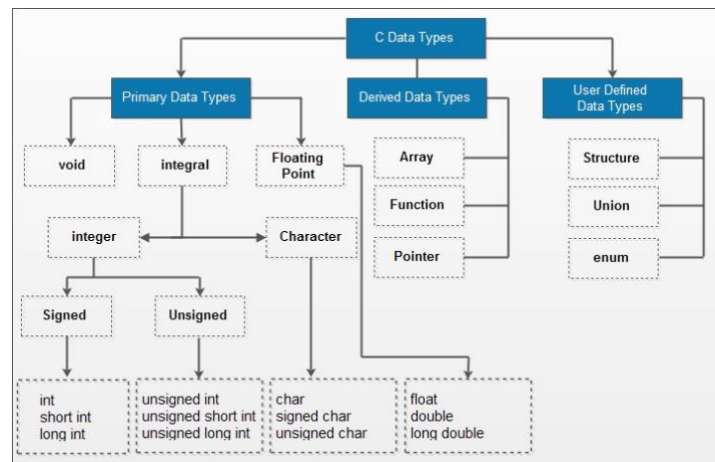
Pertimbangan desain untuk memastikan integritas berkaitan dengan tipe data, format, rentang, dan panjang:

**a. Tipe Data Primitif atau *Built-In***

Seperti *character*, *integer*, *floating-point number*, dan *boolean*.

**b. Tipe Data yang Ditentukan oleh Pemrogram (*User-Defined Data Types*)**

Namun hal ini tidak direkomendasikan dari sudut pandang keamanan karena berpotensi meningkatkan kemungkinan serangan.



Bagan tipe data pada bahasa pemrograman C.

**c. Nilai dan Operasi yang Diizinkan (*Set of Values and Permissible Operations*)**

Nilai dan operasi yang diizinkan pada kumpulan nilai, yang ditentukan oleh tipe data.

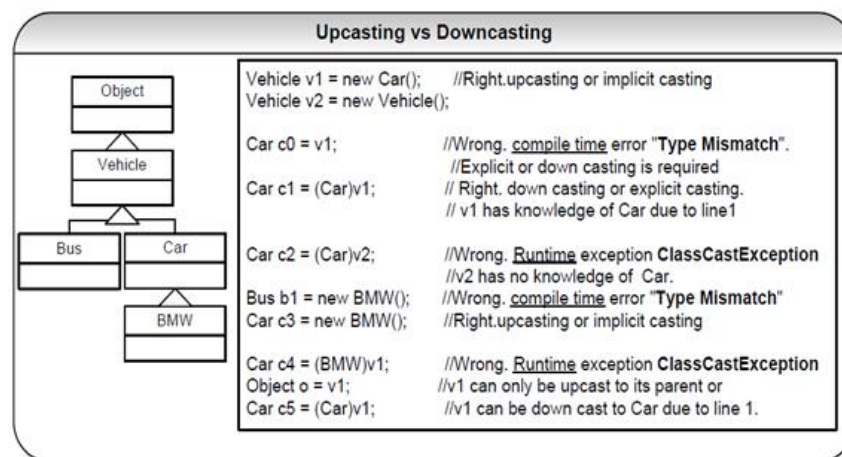
type	set of values	operators	typical expressions	
			expression	value
int	integers between $-2^{31}$ and $2^{31}-1$ (32-bit two's complement)	+ (add)	$5 + 3$	8
		- (subtract)	$5 - 3$	2
		* (multiply)	$5 * 3$	15
		/ (divide)	$5 / 3$	1
		% (remainder)	$5 \% 3$	2
double	double-precision real numbers (64-bit IEEE 754 standard)	+ (add)	$3.141 - .03$	3.111
		- (subtract)	$2.0 - 2.0e-7$	1.9999998
		* (multiply)	$100 * .015$	1.5
		/ (divide)	$6.02e23 / 2.0$	3.01e23
boolean	true or false	&& (and)	true && false	false
		(or)	false    true	true
		! (not)	!false	true
		^ (xor)	true ^ true	false
char	characters (16-bit)	[arithmetic operations, rarely used]		

Daftar *set of value* dan *operator* pada bahasa pemrograman Java.

#### d. Ketidakcocokan dan kesalahan konversi (**Conversion Mismatches and Casting or Conversion Errors**)

Ketidakcocokan dan kesalahan konversi yang dapat merusak status keamanan *software*, antara lain:

- 1) Konversi melebar (*expansion*), yaitu tipe data dikonversi dari rentang ukuran yang lebih kecil ke rentang ukuran yang lebih besar. Potensi kerugian pada data adalah hilangnya presisi.
- 2) Konversi menyempit (*truncation*), yaitu tipe data dikonversi dari rentang ukuran yang lebih besar ke rentang ukuran yang lebih kecil. Potensi kerugian pada data adalah kehilangan data jika nilai yang disimpan dalam tipe data baru lebih besar dari kisaran yang diizinkan.



Ilustrasi konversi pada bahasa pemrograman Java.

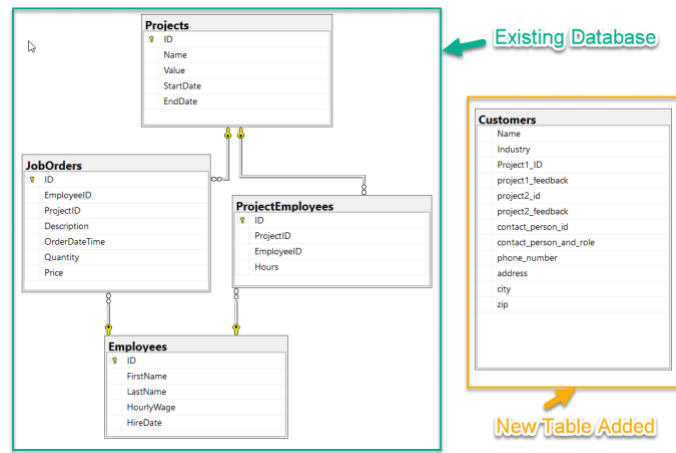
### 3.2.3 Keamanan *Database*

Desain Keamanan *database* bertujuan untuk merancang keandalan, ketahanan, dan pemulihan *software* yang bergantung pada data yang disimpan dalam *database*. Dampak desain keamanan *database* adalah:



### c. Normalisasi (Normalization)

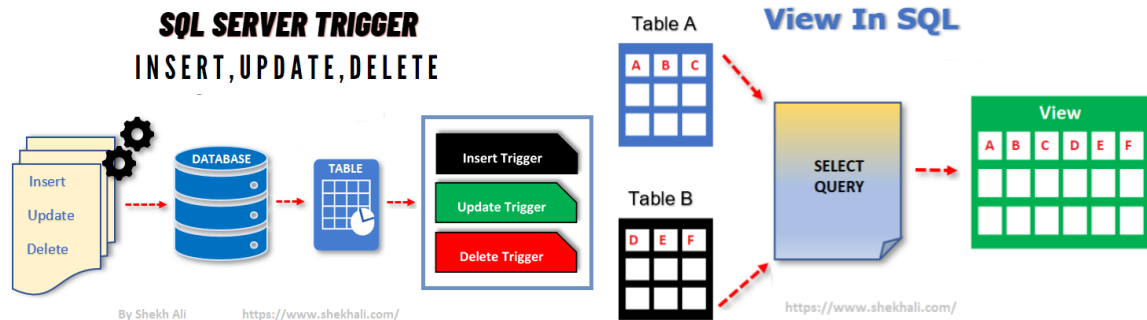
Normalisasi yaitu teknik formal yang digunakan untuk mengatur data sehingga redundansi dan inkonsistensi dapat dihilangkan.



Ilustrasi proses *database normalization* pada SQL.

### d. Trigger dan View

*Trigger* akan dipicu untuk dijalankan secara implisit pada *database* ketika peristiwa *trigger* terjadi. *Trigger* juga sangat berguna untuk mengotomatisasi dan meningkatkan mekanisme perlindungan keamanan. Sedangkan *view* adalah *output* dari *query* dan serupa dengan tabel virtual atau *query* tersimpan. *View* dibangun secara dinamis yang menyebabkan data yang disajikan dapat dibuat khusus untuk pengguna berdasarkan hak dan keistimewaannya.



Ilustrasi *trigger* dan *view* pada *database*.

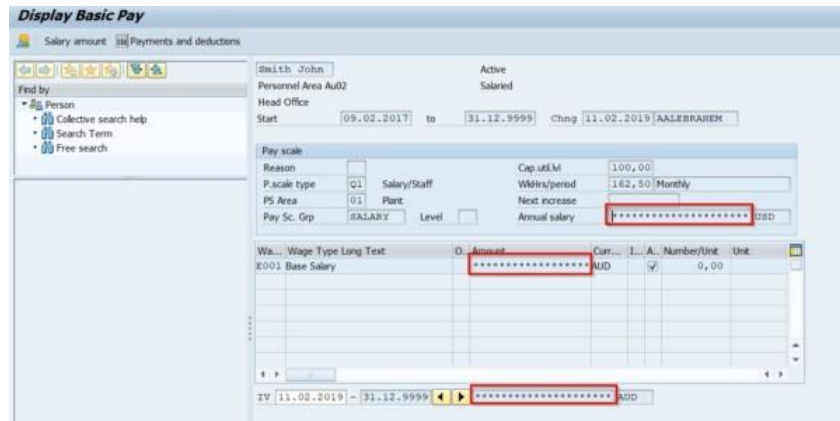
### 3.2.4 Desain Antarmuka

Pertimbangan desain antarmuka yang dapat diterapkan saat membangun *software*, antara lain:

#### a. Antarmuka Pengguna (*User Interface/UI*)

UI ini harus mendukung model keamanan dan bertindak sebagai program perantara. Desain UI harus menjamin perlindungan dari pengungkapan informasi. Penyamaran informasi sensitif, seperti *password* atau nomor kartu kredit dengan menampilkan tanda bintang di layar, adalah contoh UI yang

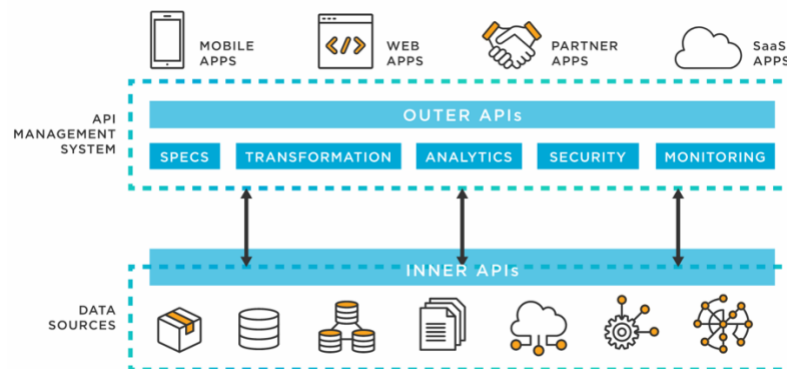
aman, yang menjamin aspek kerahasiaan. Tampilan *database* juga dapat dikatakan sebagai contoh UI yang harus dibatasi.



Ilustrasi penyamaran informasi sensitif pada UI.

## b. Antarmuka Pemrograman Aplikasi (*Application Programming Interface/API*)

API digunakan untuk berkomunikasi dari satu komponen *software* dengan yang lain atau agar *software* berinteraksi dengan sistem operasi yang mendasarinya. API biasanya tersedia di *library*, dan mungkin termasuk spesifikasi untuk rutinitas, struktur data, kelas objek, dan variabel.



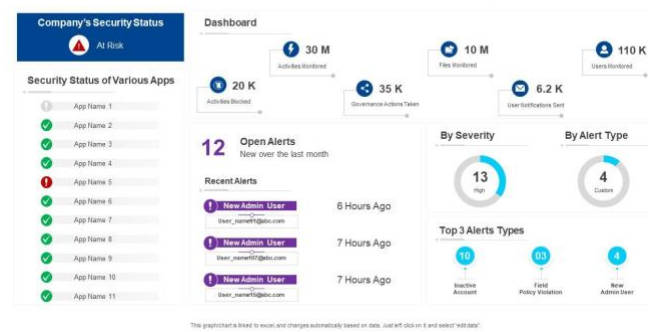
Ilustrasi API pada *software*.

## c. Antarmuka Manajemen Keamanan (*Security Management Interface/SMI*)

SMI digunakan untuk mengonfigurasi dan mengelola keamanan *software* itu sendiri, dan merupakan antarmuka administratif dengan hak istimewa (*privilege*) tertinggi. SMI digunakan untuk tugas penyediaan pengguna seperti menambahkan pengguna, menghapus pengguna, mengaktifkan atau menonaktifkan akun pengguna, serta memberikan hak dan hak istimewa untuk peran, mengubah pengaturan keamanan, mengonfigurasi pengaturan *log* audit, dan jejak audit (*audit trail*), *exception log*, dan lainnya.



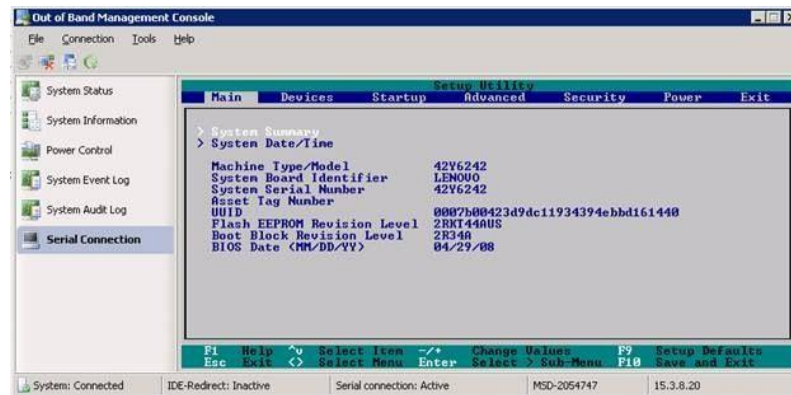
## Cloud App Security Dashboard for an Enterprise



Ilustrasi *security management interface*.

### d. Antarmuka *Out-of-Band*

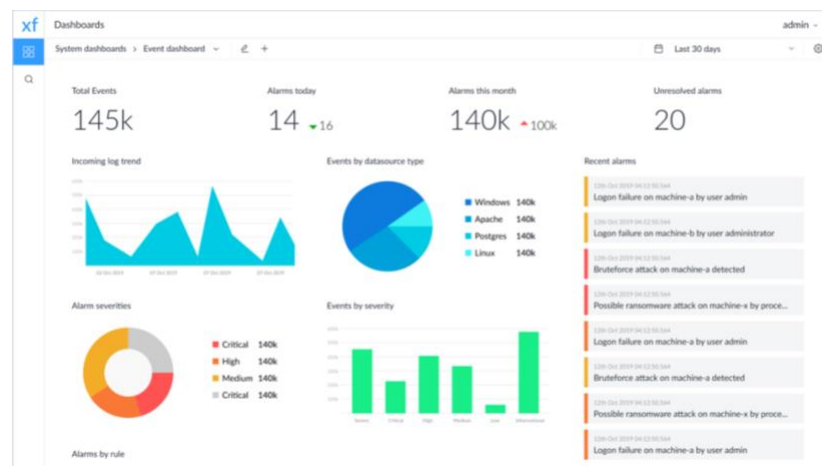
Antarmuka *out-of-band* memungkinkan administrator untuk terhubung ke komputer yang dalam keadaan *inactive* atau *shutdown*.



Ilustrasi antarmuka *out-of-band*.

### e. Antarmuka *Log*

Antarmuka *log* untuk masuk atau keluar di lingkungan yang berbeda (pengembangan, pengujian, produksi), yang merupakan komponen penting dalam audit dan saat merancang *software* untuk audit.



Ilustrasi antarmuka *log*.

### 3.2.5 Interkonektivitas

Desain interkonektivitas bertujuan untuk secara eksplisit merancang kompatibilitas *software* hulu dan hilir yang penting dalam hal pendelegasian kepercayaan, *single sign-on* (SSO), autentikasi berbasis token, dan pembagian kunci kriptografis antar aplikasi. Kompatibilitas *software* hulu dan hilir harus secara eksplisit dirancang.

Di sebagian besar aplikasi *mobile*, perlindungan data pada *client* diserahkan kepada aplikasi itu sendiri, sehingga aplikasi harus dirancang untuk menghindari penyimpanan informasi sensitif apa pun di lingkungan *sandbox* aplikasi itu sendiri. *Publisher* dan API pihak ketiga yang menyediakan layanan kriptografi (enkripsi dan dekripsi) mungkin harus dipertimbangkan untuk memastikan aspek kerahasiaan.

Saat data disimpan di suatu lokasi di jaringan, seperti pada *Network-Attached Storage* (NAS) juga harus dilindungi. Harus dirancang agar NAS hanya dapat diakses oleh koneksi yang telah diautentikasi dan diotorisasi, dibatasi oleh rentang IP tertentu, dan jika perlu telah mengakomodir perlindungan terhadap ancaman IP *spoofing*.

## 3.3 Threat Modeling

*Threat modeling* bertujuan untuk mengidentifikasi, mengukur, dan mengatasi risiko keamanan yang terkait dengan *software*.

### 3.3.1 Langkah 1: Dekomposisi Software

Dekomposisi *software* bertujuan untuk mendapatkan pemahaman tentang *software* dan bagaimana *software* berinteraksi dengan entitas eksternal. Hal ini melibatkan pembuatan *use case* untuk memahami bagaimana aplikasi digunakan, mengidentifikasi titik entri untuk melihat di mana penyerang potensial dapat berinteraksi dengan aplikasi, mengidentifikasi aset (item/area penyerang akan tertarik untuk menyerang) dan mengidentifikasi tingkat kepercayaan yang mewakili hak akses yang akan diberikan oleh *software* kepada entitas eksternal. Hal-hal yang perlu dipertimbangkan saat melakukan dekomposisi *software*:

- a. Ketergantungan eksternal.
- b. Titik entri (vektor serangan).
- c. Aset.
- d. Menentukan *attack surface* dengan menganalisis *input* (*browser input*, *cookie*, *file* properti, proses eksternal, umpan data, respons layanan, *flat file*, parameter *command line*, variabel lingkungan), *data flow*, dan transaksi.
- e. Tingkat kepercayaan (*trust level*).
- f. Analisis *data flow*.
- g. Analisis transaksi (area yang dicakup adalah validasi data/masukan data dari semua sumber yang tidak dipercaya, autentikasi, manajemen sesi, otorisasi, kriptografi pada *data at rest* dan *data in transit*, *error handling*/kebocoran informasi, *logging/auditing*).
- h. Diagram *data flow*.



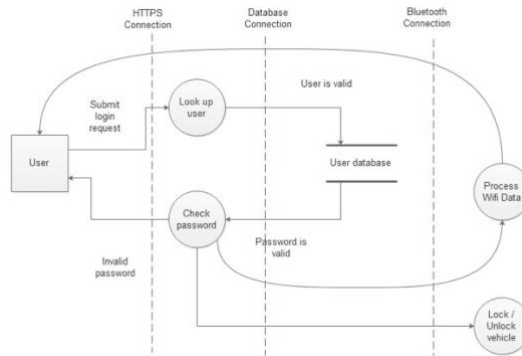
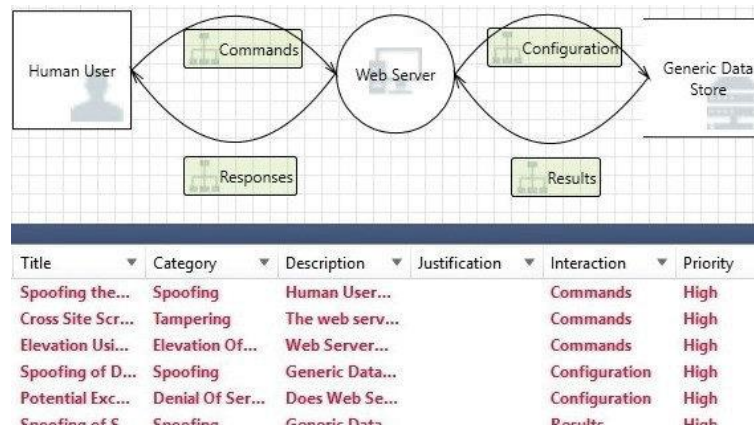


Diagram *data flow* pada dekomposisi *software*.

### 3.3.2 Langkah 2: Menentukan dan Mengurutkan Ancaman

Tahap ini bertujuan untuk mengidentifikasi ancaman menggunakan metodologi kategorisasi ancaman. Contoh kategorisasi ancaman seperti *Spoofing*, *Tampering*, *Repudiation*, *Information disclosure*, *Denial of service* and *Elevation of privilege* (STRIDE) dapat digunakan untuk mendefinisikan kategori ancaman sehubungan dengan tujuan penyerang, seperti audit dan *logging*, autentikasi, otorisasi, manajemen konfigurasi, pelindungan *data in storage* dan *data in transit*, validasi data, dan manajemen *exception*.

Untuk menentukan peringkat ancaman dapat menggunakan kriteria manajemen risiko teknologi informasi yang diterapkan di organisasi. Metodologi ini akan menentukan peringkat ancaman sesuai dengan kategori tingkat risiko.



Ilustrasi penggunaan *tools* untuk mengurutkan ancaman berdasarkan STRIDE.

### 3.3.3 Langkah 3: Menentukan Penanggulangan dan Mitigasi

Tahap ini bertujuan untuk mengurangi kerentanan dengan penerapan penanggulangan sebagai pelindungan terhadap ancaman. Penanggulangan tersebut dapat diidentifikasi menggunakan daftar pemetaan ancaman-penanggulangan. Setelah peringkat risiko ditetapkan untuk ancaman, maka ancaman dapat diurutkan dari risiko tertinggi ke risiko terendah, dan memprioritaskan upaya mitigasi, seperti menanggapi ancaman tersebut dengan menerapkan tindakan pencegahan yang teridentifikasi.

## Bab IV - Pengembangan Keamanan (*Security Development*)

Tahap pengembangan keamanan bertujuan untuk:

- Menerapkan aspek teknologi dan proses pada penulisan kode yang aman.
- Memastikan kontrol keamanan yang ditentukan dalam persyaratan keamanan telah diterapkan di dalam *code*.
- Memastikan ancaman yang teridentifikasi dari pemodelan ancaman telah dihindari selama pengkodean.

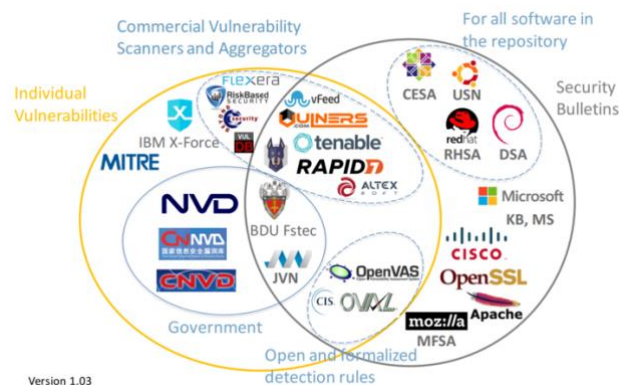
### 4.1 Identifikasi Kerentanan dan Penerapan Kontrol Umum pada *Software*

Tahap ini bertujuan untuk:

- Untuk mengidentifikasi kerentanan paling umum yang dihasilkan dari pengkodean yang tidak aman.
- Untuk menerapkan kontrol keamanan di dalam kode untuk melawan dan menggagalkan tindakan dari agen ancaman.

#### 4.1.1 Penerapan *Database* Kerentanan (*Vulnerability Database*)

Tahap ini bertujuan untuk menemukan *database* kerentanan atau repositori dari kerentanan yang diketahui yang telah ditemukan pada *software* yang diimplementasikan (misalnya, cacat/*flaw* dan *bug*). Kerentanan dan risiko keamanan *software* yang paling umum harus dicantumkan dalam *checklist secure* SDLC untuk memudahkan pengembang selama melakukan pengembangan. Namun, organisasi dapat mengacu pada daftar kerentanan *software* terkini yang ditemukan di seluruh industri pengembangan *software*, seperti OWASP Top 10 Project atau Common Weakness Enumeration (CWE/25) yang merupakan proyek yang dikelola oleh MITRE dan SANS Institute.



Kategori *database* kerentanan.

#### 4.1.2 Praktek *Secure Coding*

Tahap ini bertujuan untuk mengenali, mengevaluasi, dan mengurangi *attack surface* pada *code*. Hal ini terjadi karena *attack surface* berpotensi meningkat setiap kali 1 (satu) baris *code* ditulis. Beberapa contoh pengurangan *attack surface* yang terkait dengan *code* adalah:

- Mengurangi jumlah *code* dan layanan yang dijalankan secara *default*.
- Mengurangi volume *code* yang dapat diakses oleh pengguna yang tidak berhak.
- Membatasi kerusakan ketika *code* dieksploitasi.

Praktik dan teknik *secure coding* dapat mengacu ke OWASP *Secure Coding Practices* untuk memudahkan pengembang selama melakukan pengembangan *software*.



## 10 SECURE CODING PRACTICES



Ikhtisar praktik *secure coding*.

### 4.2 Proses Pengembangan *Software* yang Aman

Tahap ini bertujuan untuk menjamin keamanan *software* dengan melakukan proses-proses tertentu sebagai tambahan penulisan *secure code*.

#### 4.2.1. *Source Code Versioning*

Implementasi *source code versioning* bertujuan untuk memastikan bahwa tim pengembangan bekerja dengan versi *code* yang benar. *Source code versioning* memberikan kemampuan untuk:

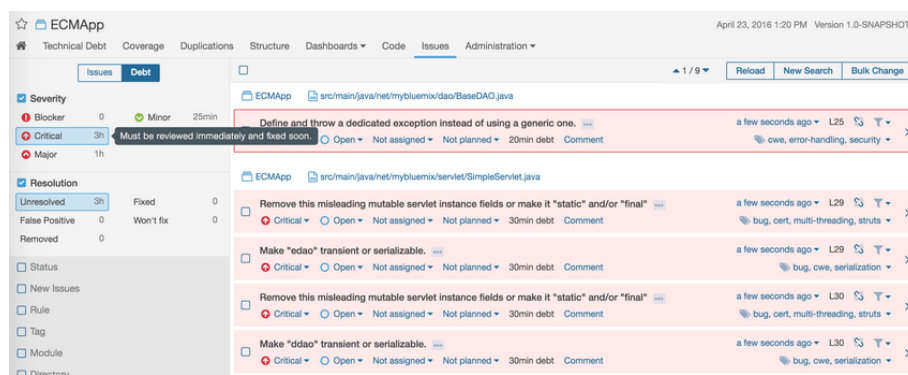
- a. *Roll back* ke versi sebelumnya jika diperlukan.
- b. Melacak kepemilikan dan perubahan *code*.
- c. Mendukung pembaruan versi.

#### 4.2.2. *Code Analysis*

Implementasi *code analysis* bertujuan untuk melakukan proses otomatis pemeriksaan *code* untuk kualitas dan kerentanan yang dapat dieksploitasi. *Code analysis* dapat dilakukan dengan 2 (dua cara):

##### a. *Static Code Analysis*

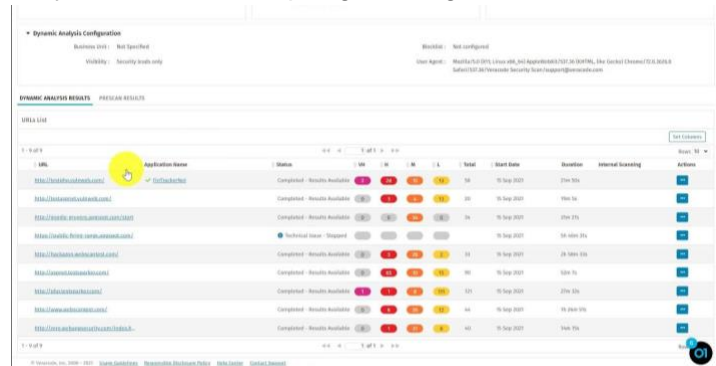
*Static code analysis* melibatkan pemeriksaan *code* tanpa menjalankan *code* atau program *software*. Analisis ini biasanya dilakukan oleh pihak ketiga.



Ilustrasi *static code analysis* menggunakan SonarCube.

### b. Dynamic Code Analysis

Dynamic code analysis adalah pemeriksaan *code* saat dijalankan sebagai program. Analisis ini biasanya dilakukan oleh pengembang.

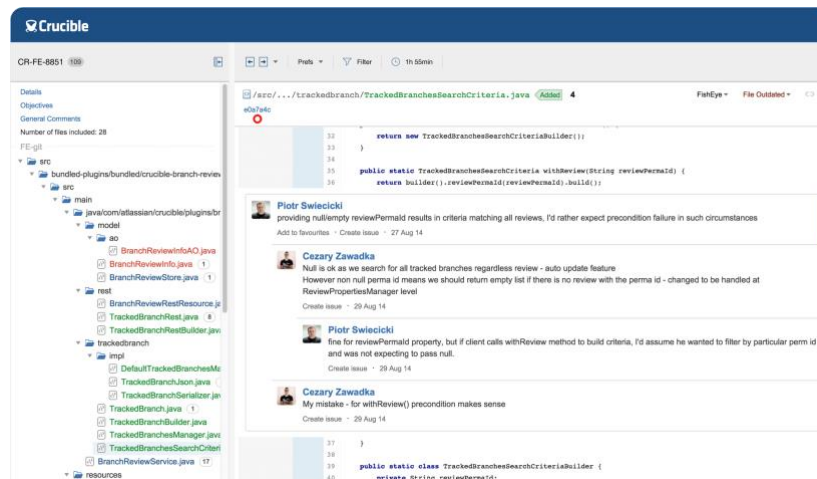


URL	Application Name	Status	Score	Start Date	Duration	Internal Scanning	Actions
https://www.veracode.com	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode/veracode/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode/veracode/veracode/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode/veracode/veracode/veracode/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode/veracode/veracode/veracode/veracode/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View
https://www.veracode.com/veracode/veracode/veracode/veracode/veracode/veracode/veracode/veracode/veracode	Veracode	Completed - Results Available	95	15 Sep 2021	2m 55s	Yes	View

Ilustrasi *dynamic code analysis* menggunakan Veracode.

### 4.2.3. Code Review

Implementasi *code review* bertujuan untuk melakukan evaluasi *source code* secara sistematis dan manual dengan tujuan menemukan masalah sintaks dan kelemahan *code* yang dapat memengaruhi kinerja dan keamanan *software*. *Code review* biasanya dilakukan oleh pengembang. Masalah semantik seperti logika bisnis dan cacat desain biasanya tidak terdeteksi dalam *code review*, tetapi *code review* dapat digunakan untuk memvalidasi *threat model* yang dihasilkan pada tahap desain.



Ilustrasi *code review* menggunakan Crucible.

### 4.2.4. Pengujian oleh Pengembang (Developer Testing)

Pengujian yang dilakukan oleh pengembang ini menggunakan alat dan teknik pengujian yang sistematis untuk membuat *software* yang diuji dapat dipelihara dengan cacat sesedikit mungkin. Pengujian oleh pengembang ini memerlukan pendekatan terstruktur dan penguasaan beberapa kompetensi inti, dimana memahami teknik pengujian mendasar dan *unit testing* adalah hal yang paling penting. Jenis pengujian umum yang dapat ditulis pengembang untuk aplikasi adalah:

**a. Pengujian Unit (*Unit Test*)**

Pengujian unit bertujuan untuk melakukan eksekusi bagian *code* atau program kecil yang diuji secara terisolasi dari *software* lengkap. Diuji secara terisolasi berarti tidak memanggil implementasi *code* yang tidak diuji, misalnya *database*, layanan *web*, atau dependensi *code* lainnya. Pengembang akan melakukan pengujian ini selama pengembangan fitur.

**b. Pengujian Integrasi (*Integration Test*)**

Pengujian integrasi bertujuan untuk melakukan eksekusi gabungan dari *code*. Dalam jenis pengujian ini, akan dilakukan pemanggilan ke *database*, layanan *web*, atau dependensi *code* lainnya. Pengembang akan melakukan pengujian ini selama pengembangan fitur.

**c. Pengujian Regresi (*Regression Test*)**

Pengujian regresi bertujuan untuk melakukan pengulangan kasus uji (*test case*) yang dijalankan sebelumnya untuk tujuan menemukan cacat pada *software* yang sebelumnya telah melewati rangkaian pengujian yang sama. Pengujian ini biasanya digunakan sebelum mengimplementasikan *code* ke lingkungan baru atau sebagai bagian dari proses pengembangan. Pengujian ini bisa menggunakan *tool* untuk otomatisasi seolah-olah pengujian dilakukan oleh tim penguji.

**d. Pengujian *Software***

Pengujian *software* bertujuan untuk melakukan eksekusi *software* dalam konfigurasi akhirnya, termasuk integrasi dengan *software* dan sistem lainnya. Pengujian ini akan menguji keamanan, kinerja, kehilangan sumber daya, permasalahan waktu, dan masalah lain yang tidak dapat diuji pada tingkat integrasi yang lebih rendah. Seperti pengujian regresi, pengujian ini bisa menggunakan *tools* untuk otomatisasi seolah-olah pengujian dilakukan oleh tim penguji.

### **4.3 Mengamankan Lingkungan Pengembangan**

Tahap ini bertujuan untuk:

- a. Melindungi *source code* agar tidak diakses oleh orang yang tidak terkait dengan proyek selama fase pengembangan.
- b. Memastikan integritas *source code* yang dikembangkan.

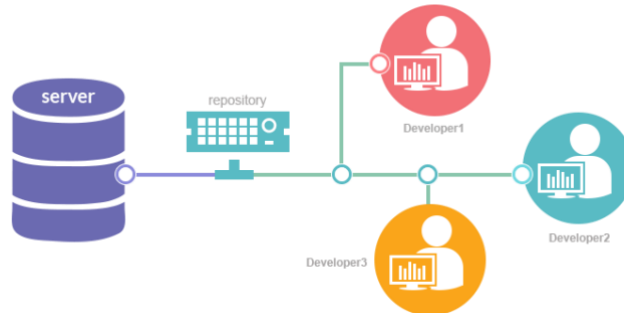
#### **4.3.1. Mengamankan Akses Fisik ke *Software* yang Membangun *Code***

#### **4.3.2. Menggunakan *Access Control List* (ACL)**

*Access Control List* (ACL) dapat mencegah akses pengguna yang tidak sah.

#### **4.3.3. Menggunakan *Software* untuk *Version Control***

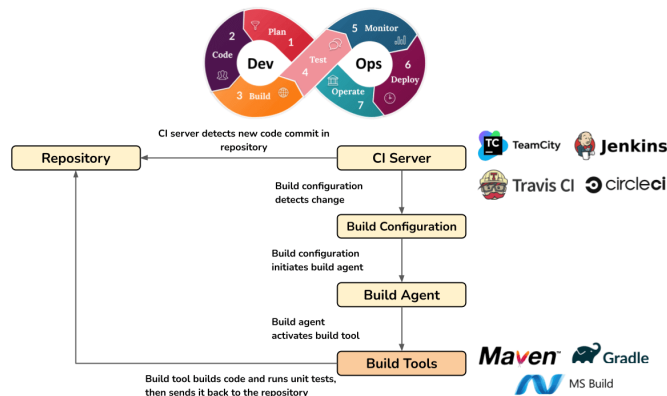
Penggunaan *software* untuk *version control* bertujuan untuk memastikan bahwa *code* yang dibangun adalah versi yang benar.



Ilustrasi penggunaan *software* untuk *version control*.

#### 4.3.4. **Build Automation**

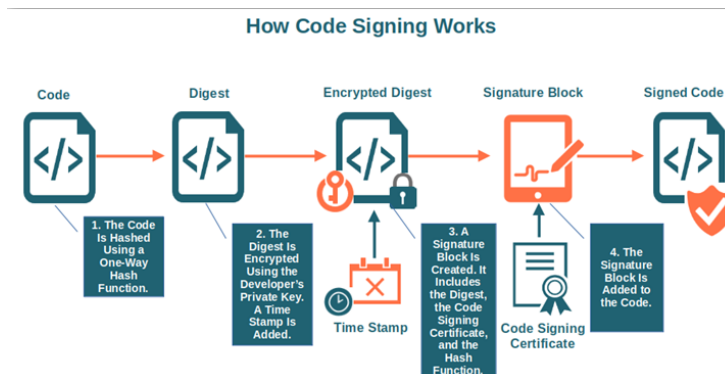
*Build automation* adalah proses pembuatan *script* atau mengotomatiskan tugas-tugas yang terlibat dalam proses *build*, namun dengan tetap membutuhkan aktivitas manual oleh tim pengembang. Beberapa aktivitas pada tahap otomatisasi *build* ini termasuk mengkompilasi *source code* menjadi *machine code*, dependensi *package*, penerapan, dan penginstalan. Saat *script* digunakan untuk membangun proses *build automation*, penting untuk memastikan bahwa kontrol dan pemeriksaan keamanan tidak dilewatkan.



Ilustrasi *build automation*.

#### 4.3.5. **Penandatanganan Code (Code Signing)**

Penandatanganan *code* bertujuan untuk memastikan integritas *source code* yang sedang dikembangkan.



Ilustrasi proses *code signing*.



## Bab V - Pengujian Keamanan (*Security Testing*)

Pengujian keamanan bertujuan untuk:

- a. Memvalidasi dan memverifikasi fungsionalitas dan keamanan *software* dengan menggunakan pengujian *quality assurance*.
- b. Memastikan *code* yang dikembangkan dapat berjalan sesuai harapan.

### 5.1 Validasi *Attack Surface*

Tahap ini bertujuan untuk memverifikasi keberadaan dan keefektifan kontrol keamanan yang dirancang dan diimplementasikan dalam *software*.

#### 5.1.1 Pengujian Pasca Pengembangan *Software*

Untuk memastikan kebenaran *software* yang dikembangkan, maka langkah ini akan mengeksekusi analisis *code* menggunakan *dynamic code analysis*, yaitu pemeriksaan *code* pada saat program dijalankan (*run as program*).

#### 5.1.2 Melakukan Pengujian Keamanan

Metode yang digunakan untuk melakukan pengujian keamanan antara lain:

##### a. Pengujian *White Box* (*White Box Testing*)

Pengujian *white box* bertujuan untuk menguji struktur *software* berdasarkan pengetahuan tentang bagaimana *software* dirancang dan diimplementasikan. Pengujian ini dikenal sebagai *full knowledge assessment* karena penguji memiliki pengetahuan lengkap tentang *software*. Pengujian ini dapat digunakan untuk menguji *use case* (perilaku yang diinginkan) serta *misuse case* (perilaku yang tidak diinginkan) dari *software* dan dapat dilakukan kapan saja setelah pengembangan *code*, meskipun disarankan untuk melakukan pengujian ini saat melakukan pengujian unit (*unit test*). Pengujian ini akan menganalisis keamanan secara metodologis dan struktural mengenai *data flow*, *control flow*, antarmuka, *trust boundary* (titik entri dan keluar), konfigurasi, *error handling* dan lainnya.

##### b. Pengujian *Black Box* (*Black Box Testing*)

Pengujian *black box* bertujuan untuk menguji perilaku *software*, juga dikenal sebagai *zero knowledge assessment* karena penguji memiliki pengetahuan yang sangat terbatas tentang *software* yang diuji. Tim penguji tidak mengetahui sama sekali mengenai dokumen arsitektur atau desain, informasi atau *file* konfigurasi, *use case*, *misuse case*, dan *source code* dari *software*. Pengujian *black box* dilakukan dengan menggunakan *tool* yang berbeda. Metodologi umum dimana pengujian *black box* dapat diselesaikan dengan *tool* adalah *fuzzing*, *scanning*, dan *penetration testing*.

##### c. Pengujian Validasi Kriptografi (*Cryptographic Validation Testing*)

Pengujian validasi kriptografi bertujuan untuk memastikan aplikasi yang menerapkan mekanisme kriptografi menggunakan algoritma kriptografi yang tepat dan *best practice* pada manajemen kunci.

#### 5.1.3 Melakukan Pengujian Keamanan *Software* untuk *Quality Assurance*

Tahap ini bertujuan untuk menerapkan berbagai jenis pengujian dan bagaimana pengujian tersebut dapat dilakukan untuk membuktikan keamanan dari *code* yang dikembangkan dalam fase pengembangan *secure SDLC*.



Pada revisi *software*, pengujian regresi harus dilakukan, dan untuk semua versi, baru atau revisi, pengujian keamanan harus dilakukan untuk memvalidasi kekuatan dari kontrol keamanan. Menggunakan daftar ancaman yang dikategorikan sebagai *template* pengujian keamanan efektif dalam memastikan cakupan komprehensif dari berbagai ancaman terhadap *software*. Idealnya, daftar ancaman yang sama yang digunakan saat *threat modeling* pada *software* akan menjadi daftar ancaman yang juga digunakan untuk melakukan pengujian keamanan. Dengan cara ini, pengujian keamanan dapat digunakan untuk memvalidasi *threat modeling*. Pengujian keamanan untuk *quality assurance*, antara lain:

- a. Pengujian validasi *input*.
- b. Pengujian untuk kontrol cacat injeksi (*injection flaw*).
- c. Pengujian untuk kontrol serangan *scripting* (*scripting attack*).
- d. Pengujian untuk kontrol nir-penyangkalan (*non-repudiation*).
- e. Pengujian untuk kontrol *spoofing*.
- f. Pengujian untuk kontrol *error* dan *exception handling* (*failure testing*).
- g. Pengujian untuk kontrol eskalasi hak istimewa (*privilege escalation*).
- h. Pengujian untuk perlindungan *anti-reversing*.
- i. Pengujian ketahanan (*stress testing*).

## 5.2 Manajemen Data Pengujian

Manajemen data pengujian bertujuan untuk mengidentifikasi *input* data pengujian dan data yang diharapkan menjadi *output* setelah operasi normal dari *software*.

### 5.2.1 Identifikasi *Output* Data Pengujian

Mengidentifikasi *output* yang diharapkan dari data pengujian bertujuan untuk mengonfirmasi apakah *software* telah memenuhi persyaratan. Kualitas data pengujian berhubungan langsung dengan kualitas pengujian itu sendiri, sehingga data pengujian perlu dikelola. Data pada tahap *production* tidak boleh diimpor dan diproses di lingkungan pengujian. Dianjurkan untuk menggunakan data tiruan dengan membuatnya dari awal di lingkungan pengujian atau simulasi.

### 5.2.2 Melakukan Pengujian dengan Transaksi Sintetis

Tahap ini bertujuan untuk melakukan transaksi dengan data *dummy* yang tidak terkait dengan proses bisnis sebenarnya. Transaksi sintetis bisa pasif atau aktif. Transaksi sintetis pasif tidak disimpan dan tidak memiliki dampak sisa pada *software* itu sendiri. Namun, jika transaksi dari pelanggan *dummy* diproses dan disimpan dalam aplikasi *software*, maka transaksi ini akan menjadi transaksi sintetis yang aktif. Penggunaan transaksi sintetis aktif mengharuskan pengembang untuk memperhatikan pengaturan data dan lingkungan sedemikian rupa sehingga tidak berdampak pada lingkungan *production*.

### 5.2.3 *Tool* pada Manajemen Data Pengujian

*Tool* pada manajemen data pengujian dapat digunakan untuk membantu dalam pembuatan *subset* data keseluruhan dari data *production* dan untuk mengurangi beberapa kekhawatiran yang menyertai pembuatan data pengujian yang berkualitas dan pengelolaannya di lingkungan pengujian. *Tool* tersebut secara otomatis menemukan hubungan data dengan menganalisis dan menangkap atribut tabel, serta memastikan

bahwa aturan ekstraksi mempertimbangkan ruang penyimpanan yang tersedia di lingkungan pengujian, sehingga proses ekstraksi tidak berakhir dengan mengekstraksi sebagian besar data yang tidak dapat diimpor ke lingkungan pengujian, karena keterbatasan ukuran.

#### **5.2.4 Pelaporan dan Pelacakan Cacat**

Tahap ini bertujuan untuk melaporkan cacat (*defect/ flaw*) dan kemudian melacak *coding bug*, cacat desain (*design flaw*), anomali perilaku (*logic flaw*), *error*, *fault*, dan kerentanan pada *software*. Pelaporan cacat harus komprehensif dan cukup rinci untuk memberikan tim pengembangan *software* berupa informasi yang diperlukan untuk menentukan akar penyebab (*root cause*) masalah sehingga dapat segera mengatasinya.

## Bab VI - Penerapan Keamanan (*Security Deployment*)

Tahap ini bertujuan untuk:

- a. Memastikan penyelesaian dari rencana operasional dan dokumentasi aplikasi.
- b. Memastikan persetujuan pihak manajemen dan penerimaan risiko untuk *deployment*.
- c. Memastikan aplikasi memenuhi fungsinya dan aman.
- d. Memastikan lingkungan dan konfigurasi untuk *deployment* yang aman.

### 6.1 Pertimbangan Penerimaan *Software*

Tahap ini bertujuan untuk:

- a. Memastikan dokumentasi aplikasi dan rencana operasional sudah siap.
- b. Mendapatkan persetujuan dan penerimaan risiko oleh pihak manajemen.

#### 6.1.1 Kriteria Penyelesaian *Software*

Tahap ini bertujuan untuk memvalidasi dan memverifikasi semua persyaratan fungsional dan keamanan sudah lengkap seperti yang diharapkan. Kriteria penyelesaian *software* pada aspek fungsionalitas dan keamanan dengan *milestone* yang jelas harus ditentukan dengan baik sebelumnya. Beberapa contoh *milestone* terkait keamanan:

- a. Pembuatan persyaratan keamanan selain persyaratan fungsional dalam tahap persyaratan.
- b. Penyelesaian *threat modeling* selama tahap desain.
- c. Reviu dan persetujuan terhadap arsitektur keamanan pada akhir tahap desain.
- d. *Code review* untuk identifikasi kerentanan setelah tahap pengembangan.
- e. Penyelesaian pengujian keamanan pada akhir tahap pengujian aplikasi.
- f. Penyelesaian dokumentasi sebelum tahap *deployment* dimulai.

#### 6.1.2 Manajemen Perubahan

Manajemen perubahan bertujuan untuk memastikan proses sudah tepat untuk menangani permintaan perubahan dan merupakan bagian dari manajemen konfigurasi. Perubahan pada lingkungan komputasi dan desain ulang arsitektur keamanan berpotensi menimbulkan kerentanan keamanan baru, sehingga dapat meningkatkan risiko. Pemberian dukungan manajemen yang diperlukan untuk *software* yang akan digunakan/dirilis harus ditetapkan.

#### 6.1.3 Persetujuan Manajemen untuk *Deployment* atau Rilis *Software*

Tahap ini bertujuan untuk memastikan semua pihak otoritas yang diperlukan untuk memberikan persetujuan. Tanpa persetujuan, maka tidak boleh ada perubahan yang diizinkan pada lingkungan *production*. Sebelum instalasi *software* baru, analisis risiko perlu dilakukan, dan risiko sisa ditentukan. Hasil analisis risiko, beserta langkah-langkah yang diambil untuk mengatasinya (mitigasi atau diterima), serta risiko residual harus dikomunikasikan kepada pemilik risiko. Persetujuan atau penolakan untuk *deploy/merilis software* harus menyertakan rekomendasi dan dukungan dari tim keamanan. Pemilik risiko harus bertanggung jawab atas persetujuan perubahan.

#### 6.1.4 Kebijakan Penerimaan dan Pengecualian Risiko

Tahap ini bertujuan untuk memastikan risiko residual dapat diterima dan/atau dilacak sebagai pengecualian jika tidak berada dalam ambang batas (*threshold*). Risiko yang tersisa setelah penerapan kontrol keamanan (risiko residual) perlu ditentukan terlebih dahulu. Pilihan terbaik untuk mengatasi risiko total adalah memitigasinya sehingga risiko residual berada di bawah ambang batas yang ditentukan, dalam hal ini risiko residual dapat diterima. Risiko harus dapat diterima oleh pemilik risiko, bukan oleh pejabat di unit kerja yang membawahi fungsi TI.

#### 6.1.5 Dokumentasi *Software*

Tahap ini bertujuan untuk memastikan semua dokumentasi yang diperlukan sudah tersedia. Mendokumentasikan *software* sangat penting untuk penggunaan *software* yang efektif, aman, dan berkelanjutan. Dokumentasi *software* harus meliputi; perancangan, penginstalan, pengaturan konfigurasi, penggunaan, dan pengaturan. Beberapa tujuan utama dokumentasi adalah untuk membuat proses penerapan *software* menjadi mudah dan dapat berulang, serta untuk memastikan bahwa operasi tidak terganggu, dan dampak terhadap perubahan *software* dapat dipahami.

### 6.2 Verifikasi dan Validasi (V&V)

Verifikasi dan validasi bertujuan untuk memastikan fungsionalitas aplikasi dan aman untuk lingkungan *production*.

#### 6.2.1 Reviu

Tahap ini bertujuan untuk melakukan reviu pada akhir setiap tahap untuk memastikan bahwa *software* berfungsi seperti yang diharapkan dan memenuhi spesifikasi bisnis yang diharapkan. Hal ini dapat dilakukan secara informal maupun formal.

#### 6.2.2 Pengujian

Tahap ini bertujuan untuk menunjukkan bahwa *software* sudah memenuhi persyaratan dan menentukan penyimpangan yang diharapkan dengan menggunakan hasil pengujian yang sebenarnya. Berbagai jenis pengujian yang dilakukan sebagai bagian dari V&V adalah:

##### a. Pengujian Deteksi Kesalahan (*Error Detection Test*)

Merupakan pengujian pada tingkat unit dan komponen. Kesalahan dapat berupa cacat (masalah desain) atau *bug* (masalah kode).

##### b. Pengujian Penerimaan (*Acceptance Test*)

Pengujian digunakan untuk menunjukkan apakah *software* siap untuk digunakan atau tidak. *Software* yang dianggap siap seharusnya tidak hanya divalidasi untuk semua persyaratan fungsional tetapi juga divalidasi untuk memastikan memenuhi persyaratan *security assurance*.

##### c. Pengujian Pihak Independen

Merupakan pengujian fungsionalitas dan *quality assurance* pada *software* dimana *software* akan direviu, diverifikasi, dan divalidasi oleh pihak lain selain pengembang *software*. Pengujian ini biasanya juga disebut sebagai verifikasi dan validasi pihak independen.

### 6.3 Sertifikasi dan Akreditasi (C&A)

Tahap ini bertujuan untuk:

- a. Mendapatkan verifikasi teknis dari fungsionalitas aplikasi.
- b. Mendapatkan penerimaan keseluruhan untuk *deployment* ke dalam lingkungan *production*.

#### 6.3.1 Sertifikasi Software

Sertifikasi *software* untuk bertujuan untuk mendapatkan verifikasi teknis pada tingkat fungsional dan *quality assurance* dari *software*. Sertifikasi keamanan mempertimbangkan *software* di lingkungan operasional. Sekurang-kurangnya, sertifikasi mencakup evaluasi *quality assurance* dari hal-hal berikut:

- a. Hak pengguna, hak istimewa, dan manajemen profil.
- b. Sensitivitas data dan aplikasi serta pengendalian yang sesuai.
- c. Konfigurasi pada *software*, fasilitas, dan lokasi.
- d. Interkonektivitas dan ketergantungan.
- e. Mode keamanan operasional.

#### 6.3.2 Akreditasi Software

Akreditasi *software* bertujuan untuk mendapatkan penerimaan formal dari pihak manajemen, sehingga diperlukan verifikasi dari pengguna yang ditargetkan terhadap *software* setelah memahami tingkat risiko *software* tersebut di lingkungan *production*. Akreditasi ini adalah keputusan resmi pihak manajemen untuk mengoperasikan *software* di mode keamanan operasional untuk jangka waktu tertentu dan merupakan penerimaan formal dari risiko yang teridentifikasi terkait dengan pengoperasian *software*.

### 6.4 Instalasi

Tahap ini bertujuan untuk mengamankan lingkungan dan konfigurasi *production* pada *software*.

#### 6.4.1 Hardening

*Hardening* bertujuan untuk mengunci *software* ke tingkat yang paling ketat sehingga dalam kondisi aman. Tingkat keamanan minimum (atau paling ketat) ini biasanya dipublikasikan sebagai *baseline* yang harus dipatuhi oleh semua *software* dalam lingkungan *production*. *Baseline* ini biasanya disebut sebagai *baseline* minimum yang dibuat berdasarkan penggunaan sistem operasi.

Penting untuk memperkuat sistem operasi *host* dengan menggunakan *baseline*, *update*, dan *patch*. Juga sangat penting untuk memperkuat *software* yang berjalan di atas sistem operasi ini. Penguatan *software* melibatkan pengaturan konfigurasi yang diperlukan dan benar dan merancang *software* agar aman secara *default*.

#### 6.4.2 Konfigurasi Lingkungan Production

Tahap ini bertujuan untuk memastikan bahwa parameter yang diperlukan untuk menjalankan *software* telah dikonfigurasi dengan tepat menggunakan *checklist* pra-instalasi. Namun demikian, masalah dinamis mungkin tidak selalu dapat diidentifikasi secara statis, maka *checklist* pra-instalasi tidak dapat memberikan jaminan bahwa *software* akan berfungsi tanpa melanggar prinsip keamanan yang dirancang dan dibuatnya. Oleh karena itu, sangat penting untuk memastikan bahwa lingkungan

pengembangan dan pengujian cocok dengan susunan konfigurasi lingkungan *production* dan pengujian simulasi secara identik meniru konfigurasi lingkungan *production* dimana *software* akan di-*deploy*.

#### **6.4.3 Manajemen Rilis**

Tahap ini bertujuan untuk memastikan rilis *software* dengan benar ke dalam lingkungan *production* setelah sumber daya *hardware* dan *software* di-*hardening* dan lingkungan dikonfigurasi untuk operasional yang aman.

Manajemen rilis adalah proses untuk memastikan bahwa semua perubahan yang dilakukan pada lingkungan komputasi telah direncanakan, didokumentasikan, diuji secara menyeluruh, dan disebar dengan hak istimewa paling sedikit (*least privilege*) tanpa berdampak negatif terhadap operasional bisnis, pelanggan, *end user*, atau tim *customer support* yang ada.

#### **6.4.4 Bootstrap dan Startup yang Aman**

Tahap ini bertujuan untuk menentukan bahwa proses pengaktifan *software* sama sekali tidak berdampak negatif terhadap kerahasiaan, integritas, atau ketersediaan *software* setelah penginstalan *software*. *Power-On Self-Test* (POST) adalah langkah pertama dalam *Initial Program Load* (IPL) dan merupakan kejadian yang perlu dilindungi agar tidak dirusak sehingga *Trusted Computing Base* (TCB) dapat dipelihara.

## Bab VII - Pemeliharaan Keamanan (*Security Maintenance*)

Tahap ini bertujuan untuk:

- a. Memantau dan menjamin bahwa *software* akan terus berfungsi dengan cara yang andal, tangguh, dan dapat dipulihkan.
- b. Mengidentifikasi *software* dan kondisi dimana *software* perlu dihapus atau diganti.

### 7.1 Operasional, Pemantauan dan Pemeliharaan

Tahap ini bertujuan untuk:

- a. Memberikan layanan kepada bisnis atau *end user* untuk kebutuhan operasional dan pemeliharaan *software*.
- b. Memastikan aspek penjaminan (*assurance*) pada pemrosesan *software* yang andal, tangguh, dan dapat dipulihkan.

#### 7.1.1 Pengamanan Operasional *Software*

Tahap ini bertujuan untuk memastikan keamanan operasional *software* dengan menjaga tetap aman atau menjaga tingkat ketahanan *software* di atas tingkat risiko yang dapat diterima. Tahap ini adalah penjaminan (*assurance*) bahwa *software* akan terus berfungsi seperti yang diharapkan dengan cara yang andal untuk bisnis tanpa mengorbankan kondisi keamanannya dengan memantau, mengelola, dan menerapkan kontrol yang diperlukan untuk melindungi aset informasi. Berbagai jenis kontrol keamanan operasional, antara lain:

- a. Kontrol detektif (*detective control*), yaitu kontrol keamanan yang dapat digunakan untuk membangun bukti dari riwayat tindakan pengguna dan proses pada *software*.
- b. Kontrol preventif (*preventive control*), yaitu kontrol keamanan yang mempersulit keberhasilan penyerang dengan mencegah serangan secara aktif atau proaktif.
- c. Kontrol pencegahan (*deterrent control*), yaitu kontrol keamanan yang tidak mencegah serangan, hanya bersifat pasif.
- d. Kontrol korektif (*corrective control*), yaitu kontrol keamanan yang bertujuan untuk dapat memberikan pemulihan sebagai penjaminan (*assurance*) pada *software*.
- e. Kontrol kompensasi (*compensating control*) adalah kontrol keamanan yang harus diterapkan ketika kontrol *software* yang ditentukan pada persyaratan keamanan tidak dapat dipenuhi karena kendala bisnis atau teknis yang terdokumentasi.

#### 7.1.2 Pemantauan Berkelanjutan

Tahap ini bertujuan untuk melakukan pemantauan terhadap sistem, *software*, atau proses. Penting untuk terlebih dahulu menentukan persyaratan pemantauan sebelum menerapkan solusi pemantauan. Persyaratan pemantauan perlu didefinisikan dari aspek bisnis di awal siklus hidup pengembangan *software*.

Pada persyaratan pemantauan, metrik terkait yang mengukur kinerja aktual dan operasional harus diidentifikasi dan didokumentasikan. Pengujian keamanan berkelanjutan harus dilakukan pada interval yang direncanakan atau sesuai dengan perubahan berdasarkan kebutuhan atau persyaratan.



### 7.1.3 Audit untuk Pemantauan

Tahap ini bertujuan untuk memberikan revidi dan pemeriksaan independen terhadap rekaman (*record*) dan aktivitas *software*. Audit dilakukan oleh auditor yang tanggung jawabnya meliputi pemilihan *event* yang akan diaudit pada *software*, pengaturan *audit flag* yang memungkinkan *record* dari *event* tersebut dan menganalisis jejaknya dari *event audit*. Audit harus dilakukan secara berkala dan dapat memberikan wawasan tentang keberadaan dan keefektifan kontrol keamanan dan privasi.

## 7.2 Manajemen Insiden Siber

Tahap ini bertujuan untuk mendeteksi dan memantau insiden siber berupa pelanggaran keamanan (*security breach*).

### 7.2.1 Menentukan Event, Alert, dan Insiden Siber

Tahap ini bertujuan untuk menentukan suatu insiden keamanan benar-benar terjadi atau tidak, sehingga harus ditentukan terlebih dahulu apa yang dimaksud dengan insiden siber. Kegagalan pada tahap ini dapat menyebabkan potensi kesalahan klasifikasi pada *event* dan *alert* sebagai insiden siber, dan hal ini dapat merugikan organisasi.

- Event* adalah setiap tindakan yang diarahkan pada suatu objek yang mencoba mengubah keadaan objek tersebut.
- Alert* adalah *event* yang ditandai dan perlu diteliti lebih lanjut untuk menentukan apakah kejadian dari *event* tersebut merupakan suatu insiden siber.
- Alert* dapat dikategorikan ke dalam insiden siber, dan *event* buruk dapat dikategorikan ke dalam insiden siber jika melakukan pelanggaran atau mengancam pelanggaran pada kebijakan keamanan jaringan, sistem, atau aplikasi *software*.

### 7.2.2 Identifikasi Jenis Insiden Siber

Tahap ini bertujuan untuk mengidentifikasi beberapa jenis insiden siber, seperti:

- Denial of Service* (DoS), yaitu serangan siber yang mencegah atau mengganggu pengguna yang sah untuk menggunakan jaringan, sistem, atau aplikasi *software* dengan cara menghabiskan sumber daya.
- Software* berbahaya (*malicious software*), yaitu insiden yang berkaitan dengan entitas berbahaya berbasis *software* seperti virus, *worm*, dan *trojan horse* yang berhasil menginfeksi *host*.
- Akses tidak sah (*unauthorized access*), yaitu insiden siber terkait kontrol akses dimana penyerang memperoleh akses logik atau fisik ke jaringan, sistem atau aplikasi *software*, data, atau sumber daya TI lainnya, tanpa memiliki hak eksplisit untuk melakukannya.
- Penggunaan yang tidak boleh (*inappropriate usage*), yaitu tindakan dimana penyerang melanggar penggunaan sumber daya *software* atau kebijakan yang dapat diterima (*acceptable use policy*) di organisasi.
- Insiden siber yang kompleks, yaitu insiden yang mencakup 2 (dua) atau lebih jenis insiden siber.

### 7.2.3 Proses Tanggap Insiden Siber

Tahap ini bertujuan untuk memungkinkan dan menjamin keamanan operasional organisasi dan tetap dalam bisnis. Fase utama dari proses tanggap insiden siber melibatkan persiapan, identifikasi, penahanan, perbaikan dan pemulihan, dan

pembelajaran yang diperoleh (*lesson learned*). Penjelasan singkatnya adalah sebagai berikut:

- a. Persiapan, yaitu menetapkan kemampuan tanggap insiden siber, mencegah insiden siber dengan memastikan bahwa sistem, jaringan, dan aplikasi dalam keadaan yang cukup aman.
- b. Identifikasi, yaitu menganalisis dan memvalidasi setiap insiden siber, mengikuti proses yang telah ditentukan sebelumnya dan mendokumentasikan setiap langkah yang diambil, melakukan pemrioritasan insiden, serta melaporkan insiden siber yang ditemukan ke Tim Tanggap Insiden Siber (*Computer Security Incident Response Team/CSIRT*).
- c. Penahanan, yaitu menjalankan strategi penahanan untuk meminimalkan pengaruh insiden siber terhadap sumber daya dan mencegah kerusakan.
- d. Perbaikan, yaitu melakukan perbaikan kerentanan dan kontrol keamanan untuk menghapus ancaman siber.
- e. Pemulihan, yaitu mengembalikan sistem yang terdampak ke operasional normal.
- f. Pembelajaran yang diperoleh (*lesson learned*), yaitu melakukan perubahan berdasarkan pembelajaran yang diperoleh dari insiden siber dan melakukan peningkatan yang berkelanjutan.

### **7.3 Manajemen Permasalahan**

Tahap ini bertujuan untuk menentukan dan menghilangkan akar penyebab (*root cause*) masalah dan untuk meningkatkan layanan yang diberikan *software* kepada operasional bisnis agar tidak terulang kembali.

#### **7.3.1 Pemberitahuan Insiden Siber**

Tahap ini bertujuan untuk mengidentifikasi dan memberi tahu terjadinya insiden siber atau permasalahan pada *software*.

#### **7.3.2 Analisis Akar Penyebab (*Root Cause Analysis*)**

Tahap ini bertujuan untuk menentukan penyebab masalah dengan menerapkan langkah-langkah analisis akar penyebab (*root cause analysis /RCA*). Analisis akar penyebab dilakukan untuk menentukan mengapa masalah tersebut terjadi secara berulang kali dan sistematis, hingga tidak ada lagi penyebab atau masalah yang harus dijawab.

#### **7.3.3 Penentuan Solusi**

Tahap ini bertujuan untuk menentukan solusi, baik solusi sementara atau solusi permanen yang akan diterapkan.

#### **7.3.4 Permintaan Perubahan**

Tahap ini bertujuan untuk menyertakan solusi (untuk mendukung pengguna), kesalahan yang diketahui (*known error*), memperbarui informasi masalah, informasi manajemen, dan permintaan perubahan.

#### **7.3.5 Menerapkan Solusi**

Tahap ini bertujuan untuk mengimplementasikan solusi yang teridentifikasi setelah memulai permintaan perubahan.

### 7.3.6 Memantau dan Melaporkan

Tahap ini bertujuan untuk memantau solusi untuk masalah dengan menyiapkan pelaporan.

## 7.4 Manajemen Perubahan

Tahap ini bertujuan untuk:

- a. Menentukan pemulihan dan penyelesaian masalah setelah akar penyebab masalah teridentifikasi.
- b. Melacak kerentanan dan memantau penyelesaian masalah untuk memastikan bahwa solusinya sudah efektif dan masalah tidak terjadi lagi.

### 7.4.1 Manajemen *Patch* dan Kerentanan

Tahap ini bertujuan untuk memperbarui atau memperbaiki *software* yang ada dengan *patch*, yaitu potongan *code* yang digunakan agar *software* tidak rentan terhadap *bug*. *Patching* adalah proses penerapan pembaruan atau perbaikan ini. *Patch* dapat digunakan untuk mengatasi masalah keamanan pada *software* atau sekadar menyediakan fungsionalitas tambahan dan merupakan bagian dari *hardening*. Beberapa langkah penting yang perlu diambil sebagai bagian dari proses *patch* meliputi:

- a. Memberi tahu pengguna *software* atau sistem tentang *patch*.
- b. Menguji *patch* dalam lingkungan yang disimulasikan sehingga tidak ada masalah dengan kompatibilitas sebelumnya atau ketergantungan (hulu atau hilir).
- c. Mendokumentasikan perubahan beserta rencana *roll back*. Perkiraan waktu untuk menyelesaikan penginstalan *patch*, kriteria untuk menentukan keberhasilan *patch*, dan rencana *roll back* harus dimasukkan sebagai bagian dari dokumentasi.
- d. Mengidentifikasi masa pemeliharaan atau kapan waktu pemasangan *patch* harus dilakukan. Waktu terbaik untuk memasang *patch* adalah saat ada sedikit gangguan pada operasi normal bisnis, tetapi dengan sebagian besar *software* beroperasi dalam kondisi normal. Mengidentifikasi waktu terbaik untuk menerapkan *patch* merupakan tantangan saat ini.
- e. Memasang *patch*.
- f. Menguji *patch* setelah pemasangan di lingkungan *production* juga diperlukan. Kadang-kadang *reboot* atau *restart software* dimana *patch* diinstal diperlukan untuk membaca atau memuat pengaturan konfigurasi dan perbaikan yang lebih baru untuk diterapkan. Validasi kompatibilitas sebelumnya dan dependensi juga perlu dilakukan.
- g. Memvalidasi bahwa *patch* tidak menurunkan status keamanan dan mengoperasikan sistem dan *software* sesuai dengan *baseline* keamanan minimum.
- h. Pemantauan sistem yang di-*patch* sehingga tidak ada efek samping yang tidak diharapkan saat pemasangan *patch*.
- i. Melakukan analisis *post-mortem* jika *patch* harus dibatalkan dan menggunakan pelajaran yang didapat untuk mencegah masalah di masa mendatang. Jika *patch* berhasil, maka *baseline* keamanan minimum perlu diperbarui.
- j. Melakukan *virtual patching* untuk meng-cover sistem *legacy*. *Patch* virtual mengacu pada pembentukan lapisan *security policy enforcement* langsung yang mencegah

eksploitasi kerentanan yang diketahui, tanpa mengubah *source code* aplikasi, perubahan biner, *restart* aplikasi. *Patch* virtual dapat digunakan untuk menambahkan perlindungan sementara, memberikan waktu bagi tim pengembang untuk menerapkan *patch* fisik sesuai dengan jadwal pembaruan. *Patch* virtual juga dapat digunakan secara permanen untuk sistem lawas yang mungkin tidak dapat di-*patch*.

#### 7.4.2 Pencadangan, Pemulihan, dan Pengarsipan

Tahap ini bertujuan untuk memastikan operasional dan kelangsungan bisnis tidak terganggu. Kelangsungan bisnis tanpa gangguan merupakan faktor penting dalam pengoperasian *software* yang aman. Tidak hanya data yang harus tersedia tetapi juga sistem itu sendiri.

Selain pencadangan terjadwal secara rutin, saat *patch* dan pembaruan *software* dibuat, disarankan untuk melakukan pencadangan penuh atas *software* yang sedang diubah. Penting juga untuk memverifikasi integritas dan pemulihan cadangan (terutama cadangan data).

Selain itu, ketika *software* telah terinfeksi oleh *malware*, maka satu-satunya pilihan yang tersisa adalah untuk memastikan integritas yang berkelanjutan dengan memformat dan menginstal ulang *software* sepenuhnya disertai dengan memulihkan data dari cadangan yang aman, tepercaya, dan terverifikasi.

Arsip dapat berguna dalam dukungan pengguna, terutama untuk pelanggan sebelumnya. Integritas pada arsip dapat dicapai dengan menggunakan *hashing* dan manajemen kunci yang tepat agar data yang dilindungi secara kriptografis pada arsip dapat digunakan saat melakukan pemulihan.

### 7.5 Penghapusan

Tahap ini bertujuan untuk mengidentifikasi *software* yang tidak beroperasi untuk dihapus guna mengurangi risiko residual.

#### 7.5.1 Kebijakan Akhir Masa Pakai (*End-of-Life*)

Pada tahap ini dilakukan aktivitas manajemen risiko pada komponen sistem yang akan dihapus atau diganti untuk memastikan bahwa *hardware* dan *software* dihapus dengan benar. Untuk mengelola risiko selama fase pembuangan, penting bagi kita untuk memiliki kebijakan Akhir Masa Pakai (*End-of-Life/EOL*) yang harus ditetapkan dan dipatuhi. Kebijakan EOL adalah persyaratan pertama dalam penghapusan *software* yang aman serta data dan dokumen terkaitnya.

#### 7.5.2 Kriteria Pembuangan (*Sunset Criteria*)

Tahap ini bertujuan untuk menghapus atau mengganti produk seperti *software* atau *hardware* yang menjalankan *software* dengan mengikuti kriteria penghapusan (*sunset criteria*) sebagai pedoman.

#### 7.5.3 Proses Penghapusan (*Sunset Process*)

Tahap ini bertujuan untuk menghapus teknologi terkait *software* yang dianggap tidak aman, tetapi tidak memiliki cara untuk mengurangi risiko ke tingkat yang dapat diterima, sehingga *software* harus dihentikan sesegera mungkin.

Sesuai dengan kebijakan EOL organisasi, proses EOL yang sesuai harus ditetapkan. Proses EOL adalah serangkaian tonggak dan aktivitas teknis dan bisnis, yang ketika selesai, membuat *hardware* atau *software* menjadi usang dan tidak lagi diproduksi, dijual, diperbaiki, dipelihara, atau didukung.

#### **7.5.4 Penghapusan Informasi dan Sanitasi Media**

Tahap ini bertujuan untuk memastikan penjaminan (*assurance*) *software* dipertahankan dengan mengikuti langkah-langkah penghapusan *software* dan juga untuk memastikan bahwa media yang menyimpan informasi juga disanitasi atau dimusnahkan sebagaimana mestinya. Sanitasi adalah proses menghapus informasi dari media sehingga pemulihan dan pengungkapan data tidak mungkin dilakukan. Proses ini juga mencakup penghapusan label rahasia, tanda, dan *log* aktivitas yang terkait dengan informasi tersebut.

## Checklist Secure SDLC

### A. Checklist Persyaratan Keamanan (Security Requirement)

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
1.1	<b>Persyaratan Keamanan Inti (Core Security Requirement)</b>			
1.1.1	<b>Identifikasi Persyaratan Keamanan Inti</b>			
	a. Persyaratan kerahasiaan ( <i>confidentiality</i> )	1) Apakah diterapkan kerahasiaan pada siklus hidup informasi: <ul style="list-style-type: none"> <li>a) <i>Data in transit</i>?</li> <li>b) <i>Data in process</i>?</li> <li>c) <i>Data at rest</i>?</li> </ul> 2) Apakah diterapkan kriptografi, dengan: <ul style="list-style-type: none"> <li>a) Mekanisme terbuka (<i>overt</i>): enkripsi, hash?</li> <li>b) Mekanisme rahasia (<i>covert</i>): steganografi, <i>digital watermarking</i>?</li> </ul> 3) Apakah diterapkan penyamaran ( <i>masking</i> )?		
	b. Persyaratan integritas ( <i>integrity</i> )	Apakah terdapat diterapkan pelindungan untuk memastikan: <ul style="list-style-type: none"> <li>1) Integritas sistem?</li> <li>2) Integritas data?</li> </ul>		
	c. Persyaratan ketersediaan ( <i>availability</i> )	Apakah diterapkan pelindungan dari: <ul style="list-style-type: none"> <li>1) Upaya penghancuran <i>software/data</i>?</li> <li>2) Serangan <i>Denial of Service (DoS)</i>?</li> </ul>		
	d. Persyaratan autentikasi	Apakah diterapkan autentikasi untuk memastikan keabsahan dan memvalidasi identitas pada: <ul style="list-style-type: none"> <li>1) <i>End user</i>?</li> <li>2) Administrator?</li> <li>3) <i>Super user</i> (jika ada)?</li> </ul>		
	e. Persyaratan otorisasi	Apakah diterapkan kontrol akses untuk memastikan otoritas entitas yang terautentikasi pada sumber daya?		
	f. Persyaratan akuntabilitas	Apakah diterapkan fungsi untuk membangun catatan riwayat tindakan pengguna ( <i>audit trail</i> )?		
1.1.2	<b>Identifikasi Persyaratan Umum</b>			
	a. Persyaratan manajemen sesi	Apakah diterapkan <i>session identifier</i> untuk melacak perilaku pengguna dan mempertahankan status terautentikasi?		

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
	b. Persyaratan manajemen <i>error</i> dan <i>exception</i>	Apakah diterapkan pelindungan terhadap informasi <i>error</i> dan <i>exception</i> ?		
	c. Persyaratan Manajemen Parameter Konfigurasi	Apakah diterapkan pelindungan terhadap informasi parameter dan <i>code</i> konfigurasi yang menyusun <i>software</i> ?		
<b>1.1.3</b>	<b>Identifikasi Persyaratan Operasional</b>			
	a. Persyaratan Lingkungan Penerapan	Apakah lingkungan dimana <i>software</i> akan di- <i>deploy</i> telah diidentifikasi dan dianalisis?		
	b. Persyaratan Pengarsipan	Apakah terdapat kebutuhan pengarsipan untuk mendukung kelangsungan bisnis atau untuk memenuhi persyaratan peraturan?		
	c. Persyaratan Anti Pembajakan ( <i>Anti-Piracy</i> )	Apakah diterapkan pelindungan anti-pembajakan ( <i>anti-piracy</i> ) pada <i>software</i> yang akan dikomersialkan?		
<b>1.1.3</b>	<b>Identifikasi Persyaratan Lain</b>			
	a. Persyaratan Urutan dan Waktu	Apakah terdapat pelindungan terhadap kondisi <i>race condition</i> atau serangan <i>Time of Check/Time of Use</i> (TOC/TOU)?		
	b. Persyaratan Internasional	Apakah terdapat persyaratan hukum dan teknologi secara internasional yang harus dipatuhi?		
	c. Persyaratan Pengadaan	Apakah terdapat persyaratan keamanan <i>software</i> sudah dipenuhi pada <i>software</i> yang akan dibeli dengan proses pengadaan?		
<b>1.2</b>	<b>Klasifikasi Data</b>			
<b>1.2.1</b>	<b>Tipe Data</b>			
	a. Data Terstruktur	Apakah terdapat data terstruktur ( <i>database</i> ) pada <i>software</i> ?		
	b. Data Tidak Terstruktur	Apakah terdapat data tidak terstruktur (gambar, video, <i>email</i> , dokumen, dan teks) pada <i>software</i> ?		
1.2.2	Memberi Label pada Data	Apakah terdapat pelabelan pada data sesuai aspek C-I-A?		
1.2.3	Kepemilikan dan Peran pada Data	1) Apakah pemilik data sudah terdefinisi? 2) Apakah pemilik data sudah menjalankan perannya dalam mengelola data?		



No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
1.2.4	<i>Data Lifecycle Management</i> (DLM)	Apakah terdapat prosedur dan praktik <i>Data Lifecycle Management</i> (DLM)?		
1.2.5	Persyaratan Privasi	1) Apakah terdapat pedoman yang mengatur implementasi privasi pada data? 2) Apakah kontrol privasi sudah diterapkan sesuai persyaratan?		
<b>1.3</b>	<b>Pemodelan <i>Use Case</i> dan <i>Misuse Case</i></b>			
1.3.1	Menganalisis Skenario <i>Use Case</i>	1) Apakah sudah dibuat diagram <i>use case</i> ? 2) Apakah sudah analisis terhadap skenario <i>use case</i> ?		
1.3.2	Menganalisis Skenario <i>Misuse Case</i>	1) Apakah sudah dibuat diagram <i>misuse case</i> ? 2) Apakah sudah analisis terhadap skenario <i>misuse case</i> ?		
1.3.3	Membuat Model Serangan	1) Apakah sudah diidentifikasi serangan yang relevan dengan <i>software</i> ? 2) Apakah sudah dianalisis siapa saja yang dapat menjadi sumber ancaman?		
1.3.4	Memilih Kontrol Mitigasi	Apakah sudah dianalisis kontrol keamanan untuk memitigasi serangan pada <i>software</i> ?		
<b>1.4</b>	<b>Manajemen Risiko</b>			
1.4.1	Penilaian Risiko	Apakah sudah dilakukan aktivitas penilaian risiko berikut? 1) Karakterisasi sistem 2) Identifikasi ancaman 3) Identifikasi kerentanan 4) Analisis kontrol 5) Penentuan tingkat kemungkinan 6) Penentuan tingkat dampak 7) Penentuan tingkat risiko 8) Rekomendasi kontrol 9) Dokumentasi hasil		
1.4.2	Mitigasi Risiko	Apakah sudah dilakukan aktivitas berikut? 1) Penentuan opsi mitigasi risiko 2) Penentuan strategi mitigasi risiko 3) Menentukan pendekatan pada implementasi kontrol 4) Mengkategorikan kontrol 5) Analisis biaya dan manfaat 6) Penentuan risiko residual		

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
1.4.3	Evaluasi dan Penilaian Risiko	Apakah proses evaluasi dan penilaian sudah dilakukan secara berkala?		

## B. Checklist Desain Keamanan (*Security Design*)

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
<b>2.1</b>	<b>Pertimbangan Desain Core Security</b>			
2.1.1	Desain Kerahasiaan	Apakah sudah terdapat desain aspek kerahasiaan menggunakan hal berikut? 1) Teknik kriptografi, dengan mempertimbangkan: a) Algoritma enkripsi b) Ukuran kunci c) Manajemen kunci 2) Penyamaran ( <i>masking</i> )		
2.1.2	Desain Integritas	Apakah sudah terdapat desain aspek integritas menggunakan hal berikut? 1) <i>Hashing</i> (fungsi <i>hash</i> ) 2) Integritas referensi ( <i>referential integrity</i> ) 3) Penguncian sumber daya ( <i>resource locking</i> )		
2.1.3	Desain Ketersediaan	Apakah sudah terdapat desain aspek ketersediaan menggunakan hal berikut? 1) Replikasi ( <i>replication</i> ) 2) <i>Failover</i> 3) Skalabilitas ( <i>scalability</i> )		
2.1.4	Desain Autentikasi	Apakah sudah terdapat desain untuk fungsi autentikasi?		
2.1.5	Desain Otorisasi	Apakah sudah terdapat desain untuk otorisasi menggunakan mekanisme kontrol akses berikut? 1) Direktori 2) <i>Access Control List</i> (ACL) 3) Matriks kontrol akses ( <i>Access Control Matrix</i> ) 4) Kapabilitas ( <i>Capability</i> ) 5) Kontrol Akses Berorientasi Prosedur ( <i>Procedure Oriented Access Control</i> )		
2.1.6	Desain Akuntabilitas	Apakah sudah terdapat desain untuk mendukung <i>audit trail</i> ?		

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
<b>2.2</b>	<b>Pertimbangan Desain Tambahan</b>			
2.2.1	Bahasa Pemrograman	Apakah bahasa pemrograman yang akan digunakan sudah ditentukan?		
2.2.2.	Jenis, Format, Jangkauan, dan Panjang Data	Apakah sudah ditentukan hal berikut? 1) Tipe data primitif atau <i>built-in</i> 2) Tipe data yang ditentukan oleh pemrogram ( <i>User-Defined Data Types</i> ) 3) Nilai dan operasi yang diizinkan ( <i>Set of Values and Permissible Operations</i> ) 4) Ketidakcocokan dan kesalahan konversi ( <i>Conversion Mismatches and Casting or Conversion Errors</i> )		
2.2.3	Keamanan <i>Database</i>	Apakah sudah terdapat desain keamanan <i>database</i> dengan mempertimbangkan hal berikut? 1) <i>Polyinstantiation</i> 2) Enkripsi <i>database</i> ( <i>database encryption</i> ) 3) Normalisasi ( <i>Normalization</i> ) 4) <i>Trigger</i> dan <i>view</i>		
2.2.4	Desain Antarmuka	Apakah sudah terdapat desain antarmuka pada hal berikut? 1) Antarmuka pengguna ( <i>user interface/UI</i> ) 2) Antarmuka pemrograman aplikasi ( <i>application programming interface/API</i> ) 3) Antarmuka manajemen keamanan ( <i>security management interface/SMI</i> ) 4) Antarmuka <i>out-of-band</i> 5) Antarmuka <i>log</i>		
2.2.5	Interkonektivitas	Apakah sudah terdapat desain interkonektivitas pada <i>software</i> hulu dan hilir?		
<b>2.3</b>	<b>Pemodelan Ancaman</b>			
2.3.1	Dekomposisi <i>Software</i>	Apakah sudah dilakukan dekomposisi <i>software</i> dengan mempertimbangkan hal berikut? 1) Ketergantungan eksternal 2) Titik entri (vektor serangan) 3) Aset 4) <i>Attack surface</i> 5) Tingkat kepercayaan ( <i>trust level</i> )		

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
		6) Analisis <i>data flow</i> 7) Analisis transaksi 8) Diagram <i>data flow</i>		
2.3.2	Menentukan dan Mengurutkan Ancaman	1) Apakah sudah mengidentifikasi ancaman dengan metode pengkategorian ancaman (misalnya STRIDE)? 2) Apakah sudah mengurutkan ancaman menggunakan kriteria pada manajemen risiko keamanan teknologi informasi?		
2.3.3	Menentukan Penanggulangan dan Mitigasi	1) Apakah sudah mengurutkan ancaman berdasarkan peringkat risiko? 2) Apakah sudah mengidentifikasi penanggulangan berdasarkan pemetaan ancaman-penanggulangan?		

### C. Checklist Pengembangan Keamanan (*Security Development*)

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
<b>3.1</b>	<b>Identifikasi Kerentanan dan Penerapan Kontrol Umum pada <i>Software</i></b>			
3.1.1	Penerapan <i>Database Kerentanan</i>	1) Apakah organisasi mengacu ke <i>database</i> kerentanan (OWASP Top 10 Project atau CWE/25)? 2) Apakah terdapat kerentanan yang ditemukan pada <i>software</i> yang diimplementasikan berdasarkan informasi di <i>database</i> kerentanan?		
3.2.3	Praktek <i>Secure Coding</i>	Apakah teknik <i>secure coding</i> untuk mengurangi <i>attack surface</i> sudah dipraktekan?		
<b>3.2</b>	<b>Proses Pengembangan <i>Software</i> yang Aman</b>			
3.2.1	<i>Source code versioning</i>	Apakah <i>source code versioning</i> sudah diterapkan?		
3.2.2	<i>Code analysis</i>	Apakah <i>code analysis</i> diterapkan menggunakan: 1) <i>Static code analysis</i> ? 2) <i>Dynamic code analysis</i> ?		
3.2.3	<i>Code Review</i>	Apakah dilakukan <i>code review</i> secara sistematis dan manual?		
3.2.4	Pengujian oleh Pengembang ( <i>Developer Testing</i> )	Apakah pengembang sudah melakukan pengujian menggunakan <i>tool</i> dan teknik pengujian yang sistematis yang meliputi: 1) Pengujian unit?		

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
		2) Pengujian integrasi? 3) Pengujian regresi? 4) Pengujian <i>software</i> ?		
<b>3.3</b>	<b>Mengamankan Lingkungan Pengembangan</b>			
3.3.1	Mengamankan Akses Fisik ke <i>Software</i> yang Membangun <i>Code</i>	Apakah akses fisik ke <i>software</i> yang membangun <i>code</i> sudah diamankan?		
3.3.2	Menggunakan <i>Access Control List</i> (ACL)	Apakah digunakan <i>Access Control List</i> (ACL) untuk mencegah akses pengguna yang tidak sah?		
3.3.3	Menggunakan <i>Software</i> untuk <i>Version Control</i>	Apakah <i>software</i> untuk <i>version control</i> sudah digunakan?		
3.3.4	<i>Build Automation</i>	Apakah <i>build automation</i> sudah digunakan?		
3.3.5	Penandatanganan Kode ( <i>Code Signing</i> )	Apakah <i>code signing</i> sudah digunakan?		

#### D. Checklist Pengujian Keamanan (Security Testing)

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
<b>4.1</b>	<b>Validasi Permukaan Serangan (<i>Attack Surface</i>)</b>			
4.1.1	Pengujian Pasca Pengembangan	Apakah sudah dilakukan <i>dynamic code analysis</i> , yaitu pemeriksaan kode pada saat program dijalankan?		
4.1.2	Pengujian Keamanan Menggunakan Metode Pengujian Keamanan	Apakah dilakukan pengujian keamanan menggunakan hal berikut? 1) Pengujian <i>white box</i> 2) Pengujian <i>black box</i> 3) Pengujian validasi kriptografi		
4.1.3	Melakukan Pengujian Keamanan <i>Software</i> untuk <i>Quality Assurance</i>	Apakah dilakukan pengujian keamanan <i>software</i> untuk <i>quality assurance</i> yang meliputi: 1) Pengujian validasi <i>input</i> 2) Pengujian untuk kontrol cacat injeksi ( <i>injection flaw</i> ) 3) Pengujian untuk kontrol serangan <i>scripting</i> ( <i>scripting attack</i> )		

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
		4) Pengujian untuk kontrol nir-penyangkalan ( <i>non-repudiation</i> ) 5) Pengujian untuk kontrol <i>spoofing</i> 6) Pengujian untuk kontrol <i>error</i> dan <i>exception handling</i> ( <i>failure testing</i> ) 7) Pengujian untuk kontrol eskalasi hak istimewa ( <i>privilege escalation</i> ) 8) Pengujian untuk pelindungan <i>anti-reversing</i> 9) Pengujian ketahanan ( <i>stress testing</i> )		
<b>4.2</b>	<b>Manajemen Data Pengujian</b>			
4.2.1	Mengidentifikasi <i>Output</i> Data Pengujian untuk Memastikan Persyaratan <i>Software</i>	1) Apakah <i>output</i> dari data pengujian sudah sesuai dengan harapan atau telah memenuhi persyaratan? 2) Apakah menggunakan data tiruan di lingkungan pengujian atau simulasi?		
4.2.1	Melakukan Pengujian dengan Transaksi Sintetis	Apakah transaksi pada pengujian dilakukan menggunakan data <i>dummy</i> yang tidak terkait dengan proses bisnis sebenarnya?		
4.2.3	Solusi pada Manajemen Data Pengujian	Apakah digunakan solusi ( <i>tool</i> atau layanan) pada manajemen data pengujian?		
4.2.4	Pelaporan dan Pelacakan Cacat	Apakah terdapat mekanisme untuk melaporkan cacat ( <i>defect/bug</i> ) dan kemudian melacak <i>coding bug</i> , cacat desain ( <i>design flaw</i> ), anomali perilaku ( <i>logic flaw</i> ), <i>error</i> , <i>fault</i> , dan kerentanan pada <i>software</i> ?		

### E. Checklist Penerapan Keamanan (*Security Deployment*)

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
<b>5.1</b>	<b>Pertimbangan Penerimaan <i>Software</i></b>			
5.1.1	Kriteria Penyelesaian	Apakah terdapat kriteria penyelesaian untuk fungsionalitas dan keamanan <i>software</i> dengan <i>milestone</i> yang jelas?		
5.1.2	Manajemen Perubahan	Apakah terdapat mekanisme untuk menangani permintaan perubahan pada penerapan <i>software</i> ?		

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
5.1.3	Persetujuan untuk <i>Deployment</i> atau Rilis	Apakah terdapat mekanisme persetujuan/penolakan untuk menerapkan/merilis <i>software</i> ?		
5.1.4	Kebijakan Penerimaan dan Pengecualian Risiko	Apakah terdapat kebijakan untuk penerimaan dan pengecualian risiko?		
5.1.5	Dokumentasi <i>Software</i>	Apakah terdapat dokumentasi <i>software</i> yang memadai, yang meliputi hal berikut? 1) Perancangan 2) Penginstalan 3) Pengaturan konfigurasi 4) Penggunaan 5) Pengaturan		
<b>5.2</b>	<b>Verifikasi dan Validasi (V&amp;V)</b>			
5.2.1	Reviu	Apakah dilakukan reviu pada akhir setiap tahap untuk memastikan bahwa <i>software</i> berfungsi seperti yang diharapkan dan memenuhi spesifikasi bisnis yang diharapkan?		
5.2.2	Pengujian	Apakah sudah dilakukan pengujian untuk memastikan persyaratan sudah dipenuhi dan menentukan penyimpangan yang diharapkan dengan melakukan hal berikut? 1) Pengujian deteksi kesalahan 2) Pengujian penerimaan 3) Pengujian pihak independen		
<b>5.3</b>	<b>Sertifikasi dan Akreditasi (C&amp;A)</b>			
5.3.1	Sertifikasi <i>Software</i>	Apakah dilakukan sertifikasi <i>software</i> yang mencakup evaluasi penjaminan ( <i>assurance</i> ) hal berikut? 1) Hak pengguna, hak istimewa, dan manajemen profil 2) Sensitivitas data dan aplikasi serta pengendalian yang sesuai 3) Konfigurasi pada <i>software</i> , fasilitas, dan lokasi 4) Interkoneksi dan ketergantungan 5) Mode keamanan operasional		
5.3.2	Mendapatkan Akreditasi	Apakah terdapat mendapatkan akreditasi atas penerimaan formal pihak manajemen pada penggunaan <i>software</i>		
<b>5.4</b>	<b>Instalasi</b>			
5.4.1	<i>Hardening</i>	1) Apakah dilakukan <i>hardening</i> sistem operasi <i>host</i> dengan menggunakan <i>baseline</i> , <i>update</i> , dan <i>patch</i> ?		



No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
		2) Apakah dilakukan <i>hardening</i> pada <i>software</i> melibatkan pengaturan konfigurasi dan perancangan <i>software</i> agar aman secara <i>default</i> ?		
5.4.2	Konfigurasi Lingkungan	1) Apakah sudah dipastikan bahwa lingkungan pengembangan dan pengujian cocok dengan lingkungan <i>production</i> ? 2) Apakah terdapat <i>checklist</i> pra-instalasi untuk menentukan parameter yang diperlukan dalam menjalankan <i>software</i> dengan konfigurasi yang tepat?		
5.4.3	Manajemen Rilis	Apakah terdapat mekanisme untuk memastikan rilis <i>software</i> dengan benar ke dalam lingkungan operasional?		
5.4.4	<i>Bootstrap</i> dan <i>Startup</i> yang Aman	Apakah <i>Power-On Self-Test</i> (POST) sudah dijalankan setelah instalasi <i>software</i> ?		

#### F. Checklist Pemeliharaan Keamanan (*Security Maintenance*)

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
<b>6.1</b>	<b>Operasional, Pemantauan dan Pemeliharaan</b>			
6.1.1	Pengamanan Operasional <i>Software</i>	Apakah sudah diterapkan kontrol keamanan operasional yang meliputi hal berikut? 1) Kontrol detektif 2) Kontrol preventif 3) Kontrol pencegahan 4) Kontrol korektif 5) Kontrol kompensasi		
6.1.2	Pemantauan Berkelanjutan	1) Apakah terdapat pemantauan berkelanjutan terhadap sistem, <i>software</i> , atau proses? 2) Apakah terdapat metrik terkait yang mengukur kinerja aktual dan operasional pemantauan?		
6.1.3	Audit untuk Pemantauan	Apakah audit terhadap catatan dan aktivitas <i>software</i> sudah dilakukan secara berkala?		
<b>6.2</b>	<b>Manajemen Insiden</b>			

No.	Persyaratan Keamanan	Detail Persyaratan Keamanan	Keterangan	Checklist
6.2.1	Menentukan <i>Event, Alert</i> , dan Insiden Siber	Apakah terdapat kebijakan atau prosedur yang memuat definisi tentang <i>event, alert</i> , dan insiden siber?		
6.2.2	Identifikasi Jenis Insiden Siber	Apakah terdapat kebijakan atau prosedur yang memuat tentang jenis insiden siber yang ditangani di organisasi?		
6.2.3	Proses Tanggap Insiden Siber	Apakah terdapat kebijakan atau prosedur yang memuat proses tanggap insiden siber?		
<b>6.3</b>	<b>Manajemen Permasalahan</b>			
6.3.1	Pemberitahuan Insiden Siber	Apakah terdapat prosedur yang mengatur tentang pemberitahuan insiden siber?		
6.3.2	Analisis Akar Penyebab ( <i>Root Cause Analysis</i> )	Apakah terdapat prosedur yang mengatur tentang analisis akar penyebab ( <i>root cause analysis</i> ) dari permasalahan?		
6.3.3	Penentuan Solusi	Apakah terdapat prosedur yang mengatur tentang penentuan solusi dari permasalahan?		
6.3.4	Permintaan Perubahan	Apakah terdapat prosedur yang mengatur tentang permintaan perubahan terkait permasalahan?		
6.3.5	Menerapkan Solusi	Apakah terdapat prosedur yang mengatur tentang penerapan solusi dari permasalahan?		
6.3.6	Memantau dan Melaporkan	Apakah terdapat prosedur yang mengatur tentang pelaporan permasalahan yang bertujuan memantau penerapan solusi.		
<b>6.4</b>	<b>Manajemen Perubahan</b>			
6.4.1	Manajemen <i>Patch</i> dan Kerentanan	Apakah terdapat prosedur yang mengatur tentang manajemen <i>patch</i> dan kerentanan?		
6.4.2	Pencadangan, Pemulihan, dan Pengarsipan	Apakah terdapat prosedur yang mengatur tentang pencadangan, pemulihan, dan pengarsipan?		
<b>6.5</b>	<b>Pembuangan</b>			
6.5.1	Kebijakan Akhir Masa Pakai ( <i>End-of-Life</i> )	Apakah terdapat kebijakan yang mengatur tentang <i>End-Of-Life (EOL) software</i> ?		
6.5.2	Kriteria Penghapusan ( <i>Sunset Criteria</i> )	Apakah terdapat prosedur yang mengatur tentang kriteria penghapusan <i>software (sunset criteria)</i> ?		
6.5.3	Proses Penghapusan ( <i>Sunset Process</i> )	Apakah terdapat prosedur yang mengatur tentang proses penghapusan <i>software (sunset process)</i> ?		
6.5.4	Penghapusan Informasi dan Sanitasi Media	Apakah terdapat prosedur yang mengatur tentang penghapusan informasi dan sanitasi media?		

## Referensi

- What is Secure SDLC? (<https://www.checkpoint.com/cyber-hub/cloud-security/what-is-secure-sdlc/>)
- Guidelines for Secure Software Development Life Cycle (SSDLC). Ministry of Communication and Multimedia Malaysia and CyberSecurity Malaysia.
- Sumber gambar:
  - <https://portswigger.net/web-security/race-conditions>
  - [https://blog.isc2.org/isc2\\_blog/2021/06/best-practices-and-techniques-for-pseudonymization.html](https://blog.isc2.org/isc2_blog/2021/06/best-practices-and-techniques-for-pseudonymization.html)
  - <https://wiki.cac.washington.edu/pages/viewpage.action?pageId=14850858>
  - [https://owasp.org/www-community/Threat\\_Modeling\\_Process](https://owasp.org/www-community/Threat_Modeling_Process)
  - <https://www.malwarebytes.com/blog/news/2018/03/encryption-101-how-to-break-encryption>
  - <https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences>
  - <https://jatheon.com/blog/data-at-rest-data-in-motion-data-in-use/>
  - <https://www.baeldung.com/cs/dbms-synchronization-lock-vs-latch>
  - [https://documentation.solarwinds.com/en/success\\_center/servu/content/servu-domain-directories-directory-access.htm](https://documentation.solarwinds.com/en/success_center/servu/content/servu-domain-directories-directory-access.htm)
  - <https://iotac.eu/front-end-access-control-a-new-solution-for-capability-based-access-control/>
  - <https://www.partech.nl/en/publications/2021/03/managed-and-unmanaged-code---key-differences>
  - <https://ecomputernotes.com/what-is-c/types-and-variables/data-types-in-c>
  - <https://algs4.cs.princeton.edu/11model/>
  - <https://www.expertsmind.com/questions/define-the-type-casting-3017616.aspx>
  - <https://netlibsecurity.com/white-papers/automatic-encryption/>
  - <https://www.sqlservercentral.com/articles/database-normalization-in-sql-with-examples>
  - <https://www.shekhali.com/sql-server-trigger-update/>
  - <https://www.shekhali.com/view-in-sql-server/>
  - <https://xiting.com/en/ui-masking-in-sap-gui-for-windows-part-3-of-3/>
  - <https://www.tibco.com/reference-center/what-is-an-api>
  - <https://www.geeksforgeeks.org/compare-in-band-and-out-of-band-management-access/>
  - <https://avleonov.com/2018/06/05/vulnerability-databases-classification-and-registry/>
  - <https://kinsta.com/blog/code-review-tools/>
  - <https://www.lambdatest.com/learning-hub/build-automation>
  - <https://cheapsslsecurity.com/blog/a-primer-on-how-code-signing-works/>