Sistem Keamanan pada Pengembangan Sistem Informasi

untuk *Software Developer* sektor Usaha Kecil dan Menengah (UKM)

Untuk memenuhi tugas matakuliah

Sistem Keamanan dan Proteksi

Topik Bab 7

oleh

Kelompok 125

7204000098 – Akhmad Agus

7204000217 – Dian Rahayu

7204000519 – M. Iqbal Saryuddin A.

Program Magister Teknologi Informasi

Fakultas Ilmu Komputer

Universitas Indonesia

Desember 2005

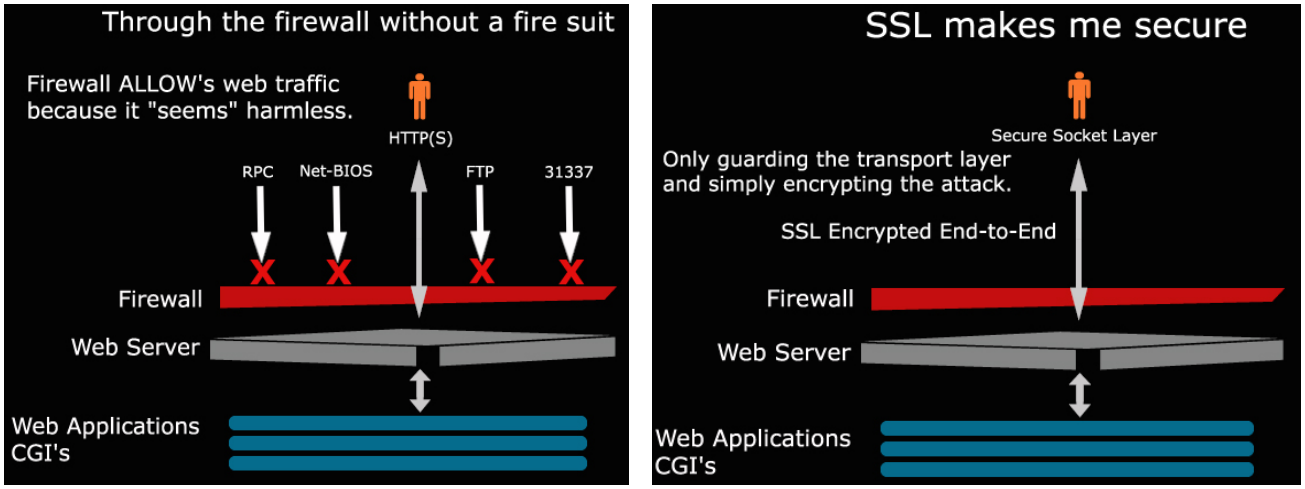
DAFTAR ISI

I. PENDAHULUAN

Menurut Curtis Coleman, MSIA, CISSP, CISM direktur Global IT Governance dari perusahaan Seagate Technology [1] mengatakan bahwa kerawanan yang paling berbahaya pada perusahaan besar yang memiliki jaringan luas adalah pada aplikasinya. Sistem keamanaan telah banyak terfokus pada antivirus dan keamanan jaringan, tapi bagian yang amat merisaukan adalah transaksi bisnis yang memiliki data yang sangat berharga (*valuable).* Sistem keamanan pada Aplikasi merupakan tren masa depan yang dapat dikatakan sebagai era baru setelah era anti Virus dan era keamanan jaringan.

Bisa dikatakan bahwa SSL (*Secure Socket Layer*) dan data enkripsi saja tidak cukup dikarenakan hal tsb hanya melindungi infomasi selama transmisi tetapi data ini tetap bisa digunakan oleh sistem yang harus diubah ke media yang bisa dibaca. Keganjilannya juga data tidak disimpan pada format ter-enkripsi dan yang lebih herannya lagi adalah data dapat dengan mudah diakses dari beberapa aplikasi dengan mudah tanpa adanya sistem keamanan yang memadai.

Selain itu juga penggunaan *firewall* tidak cukup untuk mengamankan data dan aplikasi dikarenakan Port tertentu (80 dan 443) bisa terlewati dari *firewall* seperti diilustrasikan pada gambar 1.



Source: Jeremiah Grossman, BlackHat 2001 [1]

Gambar 1 *Firewall* mengijinkan aplikasi bisa dapat diakses dan SSL membuat sistem aman

Pada hakekatnya walaupun sudah mengimplementasikan *Firewall* dan SSL, kerawanan masih tetap terbuka lebar terutama terhadap apikasi dan data, hal ini diperkuat dengan hasil pengecekan dengan AppScan bahwa lebih dari 1000 aplikasi yang telah memiliki *firewalls* dan solusi enkripsi tingkat kerawannya rata-rata 98% [1]. Menurut sumber yang sama juga mengatakan bahwa mengapa keamanan aplikasi manjadikan obyek penting diantaranya:

- **Frekuensi Kejadian:**

Dua (2) dari empat (4) web sites pada perusahaan bisnis rawan untuk dibobol

- **Tingkat penembusan (*Pervasive*)**

Tujuh puluh lima persen (75%) para *hacker* dapat menembus sampai ke level aplikasi

- **Tak terdeteksi**

Tool untuk *QA Testing* belum didesain untuk mendeteksi lubang keamanan (cacat) pada suatu aplikasi

- **Tingkat yang Membahayakan**

Ketika dieksploitasi, keamanan dapat membahayakan bisnis suatu perusahaan seperti *customer value* dan kepercayaan pelanggan yang makin menurun.

Akibat adanya cacat pada sistem keamanan suatu aplikasi maka dapat menimbulkan kerawanan bisnis yang dapat dibagi menjadi 3 level yakni:

- ***Bad Business***

US Dept. of Defense dan *Software Engineering Institute* menemukan ada rata-rata 5 dari 15 *defects* (kecacatan) pada setiap 1,000 baris program

- ***Slow Business***

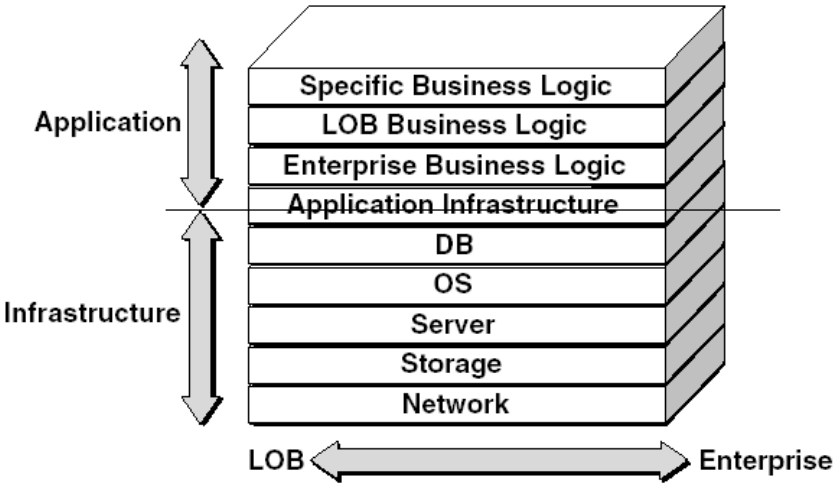
Selama 5 tahun Pentagon mempelajari bahwa diperlukan waktu 75 menit untuk menelusuri satu *defect* dan untuk menyelesaikannya diperlukan 2 sampai 9 jam.

- ***Loss of Business***

Suatu perusahaan besar dengan 1000 *servers* dapat menghabiskan $ 300.000 (tiga ratus ribu dolar) untuk mencoba dan mengimplementasikan *patch*, kebanyakan perusahaan mengimplementasikan beberapa *patch* setiap minggunya.

1.1 Apakah Sistem Keamanan pada Pengembangan Sistem Informasi

Kadang-kadang kita memang sulit untuk membedakan sistem keamanan pada level non aplikasi dengan aplikasi, mirip juga bila kita ingin membedakan infrastruktur TI dengan aplikasi seperti yang diilustrasikan pada gambar 2.



Gambar 2 Perbedaan Lapisan Infrastruktur dengan Aplikasi

Lapisan Infrastruktur IT bisa dimulai dari levek fisik (network) sampai ke level aplikasi yang masih bersifat *shareable* seperti *software components* atau *web service* yang dapat digunakan bersama oleh beberapa aplikasi yang berbeda.

Begitu juga sistem keamanan pada suatu aplikasi berbeda dengan sistem keamanan pada jaringan (era pertama yang sangat popular sampai saat ini), sistem keamanan pada *software infrastructure* (era kedua yang sedang popular juga saat ini) misalnya Anti Virus dan Sistem *backend.* Sistem keamanan pada pengembangan sistem informasi bisa juga termasuk infrastruktur aplikasi seperti database, *web services* yang ikut dikembangkan dan aplikasi bisnis sendiri.

Menurut Kepner-Tregoe [1] ada beberapa situasi yang akan dapat menimbulkan ancaman dalam pengembangan system informasi diantaranya adalah:

- Kurangnya sumber daya untuk melaksanakan prosedur dan proses ujicoba
- Tidak adanya pengukuran proses ujicoba sistem keamanan
- Kurang konsisten pada penggunaan metodologi atau proses untuk ujicoba dan kualitas
- Tidak tersedianya standard keamanan (petunjuk) untuk ujicoba
- Tidak ada satuan ukuran yang baku untuk *information assurance*
- Tidak tersedia pelatihan untuk proses ujicoba keamanan
- Tidak jelas definisi dari peran / tanggungjawab untuk aktitas ujicoba keamanan
- Tidak tersedia standard untuk *acceptance testing* untuk produk-produk yang dibeli
- Kelemahan SDLC (*Software Development Life Cicle*) yang tidak berisikan petunjuk ujicoba keamanan

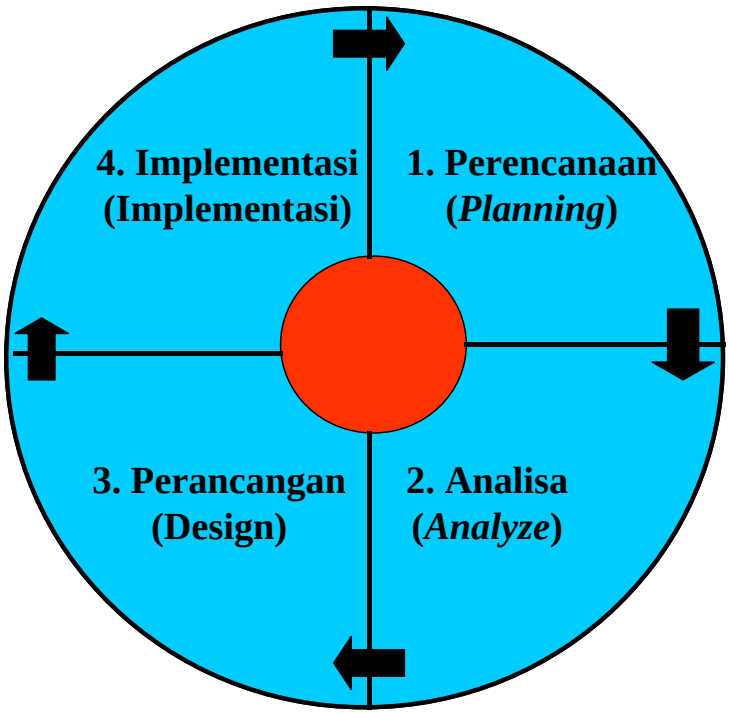
1.2 Keamanan pada Siklus Hidup Pengembangan Piranti Lunak (SDLC)

Siklus hidup pengembangan sistem informasi (aplikasi) atau sering disebut SDLC merupakan proses evolusioner yang diikuti dalam mengembangkan suatu sistem atau subsistem informasi berbasis komputer. SDLC terdiri atas serangkain tugas yang erat yang mengikuti langkah-langkah pendekatan sistem. Karena tugas-tugas tersebut mengikuti suatu pola yang teratur dan dilakukan secara top-down, SDLC sering disamakan dengan pendekatan air terjun (*waterfall approach*) walaupun pada pelaksanaannya mungkin bisa berbeda dan dapat menggunakan pendekatan lainnya.

Secara umum fase-fase dari siklus hidup pengembangan sistem informasi dapat dikelompokkan menjadi 4 fase besar [2] seperti yang diilustrasikan pada gambar 5 yakni:

- Perencanaan
- Analisa
- Design
- Implementasi

Yang masing-masing akan diuraikan lebih lanjut pada tulisan berikutnya.



Gambar 3 Fase Siklus Hidup Pengembangan Sistem Informasi (SDLC)

Dalam topik ini secara umum keamanan yang perlu diperhatikan dalam pengembangan sistem informasi dapat dikategorikan sesuai dengan fase/tahapan dalam pengembangan system informasi yakni:

- **Perencanaan**

Biasanya pada fase perencanaan ini dapat dilakukan investigasi awal dan kelayakan proyek (teknis, ekonomi dan operasional/organisasi) dan bagian kemanan yang perlu diperhatikan antara lain adalah:
-Information security policy
-Standard, legal issues
-Early validation of concepts

- **Analisa**

Pada fase ini diperlukan hal-hal berikut ini sebagai melakukan kegiatan untuk aspek keamanan: *Threat, vulnerabilities, security requirements, reasonable care, due diligence, legal liabilities, cost/benefit analysis, level of protection desired, develop test plans, validation*

- **Perancangan**

Pada fase ini juga diperlukan kegiatan-kegiatan berikut ini yang berkaitan dengan aspek keamanan yakni: *Incorporate security specifications, adjust test plans and data, determine access controls, design documentation, evaluate encryption options, design access control, consider business continuity issues, verification*

- **Implementasi**

Pada fase implementasi biasanya terkait dengan pemrograman, instalasi dan rencana pemeliharaan , adapun kegiatan-kegiatan berikut ini yang berkaitan dengan aspek keamanan yakni: *Develop information security-related code, implement unit testing, incorporate other modules or units, support business continuity plan, develop documentation.*

1.3 Sepuluh (10) besar Kerawanan pada Aplikasi

Menurut Eugene Lebanidze [3] ada 10 besar kerawanan pada aplikasi yakni:

A1. *Unvalidated Input*

Unvalidated input adalah jenis kerawanan yang cukup sering terjadi sehingga sangat serius konsekuensinya. Semua aplikasi berbasis web memerlukan penanganan masukan yang datang dari berbagai jenis sumber yang tidak dipercayai misalnya pemakai aplikasi. Jika masukan tidak divalidasi, maka pada suatu kesempatan penyerang dapat menyerang komponen aplikasi bagian belakang (database). Ada beberapa poin pada kerawanan pertama:

■ HTTP requests from browsers to web apps

Bentuk masukan dalam aplikasi Web adalah *URL, Querystring, Form Fields, Hidden Fields, Cookies, Headers*. Aplikasi web menggunakan informasi ini untuk men-*generate web pages*.

■ *Attackers* dapat mengubah semua jenis *request*

■ *Key Points:*

- Pengecekan dilakukan sebelum kamu menggunakan apasaja pada *HTTP request*
- Validasi pada sisi *client* tidak relefan
- Tolak apasaja yang tidak diijinkan

A2. *Broken Access Controls*

Akses control yang disalahgunakan dapat menimbulkan kerawaanan. Akses kontrol adalah bagaimana menjaga satu informasi pemakai (contoh sebagai otorisasi) dari jangkauan orang lain, jadi benar-benar rahasia. Poin kuncinya adalah tuliskan kebijaksanaan untuk akses control dan jangan menggunakan id yang penyerang dapat untuk memanipulasinya serta mengimplemtasikan akses control secara terpusat..

A3. *Broken Authentication and Session Management*

■ Key Points

Keep credentials secret at all times

Use only the random sessionid provided by your environment

A4. *Cross Site Scripting Flaws*

Web browser dapat menjalankan *code* yang dikirim dari *web site server* seperti Javascript atau flash sehingga memungkinkan untuk di *attack* seperti dengan mengirimkan isinya ke pemakai lain. Kata kuncinya adalah jangan membiarkan lubang untuk attacker mencuri isi web yang masih aktif.

A5. *Buffer Overflows*

Aplikasi Web membaca semua jenis masukan dari para pemakai dan biasanya bahasa pemrograman C & C++ memiliki kerawanan untuk *buffer overflows* sehingga sebaiknya jangan digunakan dan hati-hati pada saat membaca kedalam buffers. Gunakan *string* yang aman dan benar.

A6. *Injection Flaws*

Secara umum penggunaan parameter atau perintah dalam bentuk SQL jangan dilakukan dalam parameter konstan sehingga dapat diketahui oleh *attackers.*

■ Key Points
Use extreme care when invoking an interpreter
Use limited interfaces where possible (PreparedStatement)
Check return values

A7. *Improper Error Handling*

Penanganan masalah harus dilakukan dengan benar sehingga tidak meimbulkan efek lainnya yang membahayakan seperti *Out of memory, too many users, timeout, db failure,* authentication failure, access control failure, bad input. Kuncinya adalah perancangan skema penanganan *errors* yang benar dan juga konfigurasi server yang tepat.

A8. *Insecure Storage*

Penggunaan kriptografi untuk menyimpan informasi yang sensitif dan kalo bisa menggunakan algoritma yang mudah digunakan dan terintegrasi dengan baik.
Key Points
Do not even think about inventing a new algorithm
Be extremely careful storing keys, certs, and passwords
Rethink whether you need to store the information
Don't store user passwords – use a hash like SHA-256

A9. *Denial of Service*

Banyak web site diijinkan untuk melakukan administrasi secara remote sehingga membantu sekali dalam pekerjaan tapi disisi lain menjadikan kerawanan sehingga diusahakan untuk jangan menggunakan admin account untuk mengakses ke internet, pisahkan admin account

untuk aplikasi dan aplikasi utama, dan batasi *scope* untuk administrasi secara *remote* sehingga tidak terjadi DOS (*Denial of Service*)

A10. Insecure Configuration Management

Semua web dan aplikasi server memiliki pilihan konfigurasi keamanan yang relavan, seperti *default accounts* & password dan hal lain yang terkait. Oleh karena itu perlu menjaga supaya *patches* benar-benar *uptodate* dan penggunaan *scanning tools*.

1.4 Sektor UKM

Secara umum bahwa hal-hal yang terkait dengan Sistem Keamanan tersebut diatas banyak sekali berkaitan dengan perusahaan besar terutama yang mengimplementasikan *Enterprise Applications*. Bahkan menurut Kepner-Tregoe [1] menuliskan beberapa situasi yang menimbulkan kerawanan dalam sistem keamanan seperti

- Kurangnya sumber daya untuk melaksanakan prosedur dan proses ujicoba
- Tidak adanya pengukuran proses ujicoba sistem keamanan
- Kurang konsisten pada penggunaan metodologi atau proses untuk ujicoba dan kualitas
- Tidak tersedianya standard keamanan (petunjuk) untuk ujicoba
- Tidak ada satuan ukuran yang baku untuk *information assurance*
- Tidak tersedia pelatihan untuk proses ujicoba keamanan
- Tidak jelas definisi dari peran / tanggungjawab untuk aktitas ujicoba keamanan
- Tidak tersedia standard untuk *acceptance testing* untuk produk-produk yang dibeli
- Kelemahan SDLC (*Software Development Life Cicle*) yang tidak berisikan petunjuk ujicoba keamanan

Situasi tersebut diatas jelas sekali terjadi pada sektor UKM walaupun Kepner-Tregoe meriset pada sebagian banyak perusahaan-perusahaan besar. Oleh karena itu UKM sebagai *small-team* justru lebih mudah untuk melaksanakannya walaupun dengan memungkinkan terjadinya kebanjiran *jobs*/tanggungjawab pada orang yang sama. Tapi yang penting pemahaman dan daya upaya untuk perbaikan terus dilakukan.

II. KEAMANAN PADA FASE PERENCANAAN

Fase ini merupakan proses utama yang pertama kali dilakukaan untuk memahami mengapa sistem informasi sebaiknya dibuat dan kelayakannya serta dimungkinkan perlunya investigasi awal untuk mengumpulkan permasalahan-permasalahan yang ada dalam sistem yang saat ini sedang berjalan. Secara umum ada 2 tahapan yakni:

- **Tahap Kelayakan**

Tahap kelayakan untuk mengembangkan sistem informasi yang baru (pertama kali dibuat) atau menggantikan sistem informasi yang lama harus dikaji kelayakannya dari beberapa aspek, diantaranya:

· Aspek Kelayakan Teknis

Aspek kelayakan teknis ini merupakan ketersediaan Teknologi Informasi dan tenaga ahlinya sehingga bisa menjawab "*Can we build it*?"

· Aspek Kelayakan Ekonomi

Aspek kelayakan eknonomi diperlukan untuk dapat menjawab pertanyaan ini "*Will it provide business value*?" sehingga bisa meyakinkan "*project sponsor*" dan manajemen untuk menginvestasikan uangnya dalam pengembangan sistem informasi yang diusulkan.

· Aspek Kelayakan Organisasi

Aspek kelayakan ini diperlukan untuk mendapatkan dukungan sepenuhnya dari pihak organisasi baik tingkat manajemen maupun staff operasional yang akan melakukannya. Jadi intinya adalah jawaban pertanyaan ini "*If we build it, will it be used*?"

- **Tahap Investigasi Awal**

Jika manajemen sudah menyetujui bahwa pekerjaan ini dapat diteruskan untuk diproses lebih lanjut, maka tahapan berikutnya adalah fact finding yang bertujuan untuk mendapatkan informasi yang lebih detail dari proses tahapan kelayakan diatas diantaranya adalah:
-Untuk mendapatkan informasi tentang kebutuhan fungsional dari sistem yang sedang berjalan
-Untuk memperoleh kebutuhan-kebutuhan baru untuk sistem yang akan dikembangkan
-Untuk mengetahui batasan-batasan yang diminta misalnya tools yang akan dipakai
-Untuk mengetahui lingkup jenis data yang diperlukan dan volumenya
-Untuk mengetahui permasalahan-permasalahan yang ditemukan pada sistem yang sedang berjalan dan arahan-arahan yang diminta

Fakta-fakta tersebut diatas dapat diperoleh dengan beberapa cara diantaranya interview ke personal baik manajemen maupun staff operasional, penggunaan questionnaires, observasi langsung ke area aplikasi yang diinginkan, mencari dokumentasi dan contoh-contoh bentuk keluaran/laporan dan masukan untuk data entry serta proses-proses yanga tersedia.

Review Policies & Standard

Software Licensing, dalam pembuatan aplikasi nantinya harus dipastikan bahwa SELURUH entitas perangkat lunak yang terlibat dalam pembangunan aplikasi harus terjamin dan tersedia licensenya agar tidak ada kendala non teknis pada saat perangkat lunak mulai dikembangkan dan digunakan.

Business Contingency Planning, dalam perencaanaan perangkat lunak harus dapat ditentukan bagaimana desain dalam upaya menjaga realibilitas sistem aplikasi dan bagaimana rencana kontingensi jika terjadi kerusakan sehingga tidak menggangu bisnis yang sedang berjalan.

Security Policy and Procedures, Ini lebih memuat aturan yang jelas dan terdokumentasi sebagai *guideline* atau pegangan bagi seluruh personil yang terlibat dalam pengembangan aplikasi, seperti aturan network akses, user akses, waktu akses dan lainnya.

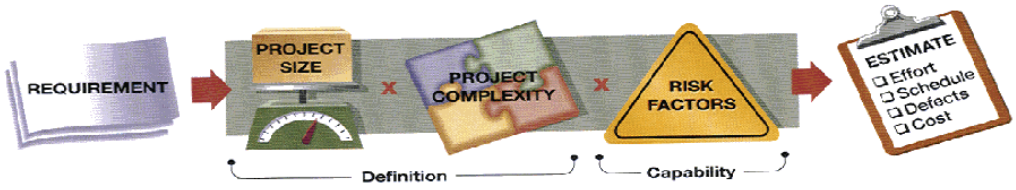
Ownership of Software Code and Related Intellectual Property, setelah aplikasi dapat diluncurkan, maka harus dapat ditegaskan kepemilikan dari perangkat aplikasi terutana code program dan bagian-bagian yang rumit dari program.

IT Project Management, perlu ada untuk menspesifikasikan kebutuhan-kebutuhan selama pelaksanaan project aktifitas didalamnya seperti Requirements Management, Project Planning, Project Tracking, Configuration Management, Risk Management, and Project Closeout.

Technical Architecture Compliance, aplikasi memiliki standar yang dapat diimplementasikan pada arsitektur teknis yang eksisting.

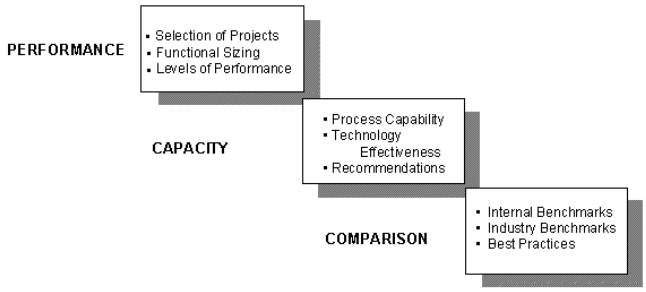
Pengembangan Pengukuran dan Kriterianya

Deliverable Sizing & Estimating, menyediakan estimasi waktu dan keakuratan untuk titik-titik kritikal dari *project*, mengukur ukuran sistem, kompleksitas perangkat lunak dan mengevaluasi faktor *critica*l



Gambar 4 Analisa Faktor Risiko

Performance Benchmarking
Benchmark performansi perlu dilakukan untuk *critical success factor* seperti *software quality*, *time to market*, dan *cost reduction*.
Kriteria yang diukur yaitu seleksi *project*, ukuran fungsi, dan tingkat performansi.



Gambar 4 *Benchmarking*

Measurement Program Development

Metode dan cara pengukuran harus juga dikembangkan dengan tujuan dapat meningkatkan nilai bisnis dan strategis. Dengan pengukuran ini akan mendapatkan niali kualitatif dan kuantitatif dari perangkat lunak dan aplikasi yang dikembangkan.

Kriteria-kriteria pengukuran :

Project Size, expressed in Function Points (FPs) or Source Lines of Code (SLOC); estimated and actual.
Business Value, net present value.
Cycle Time, project cycle time (beginning date and completion date); estimated and actual.
Delivery On Plan, slippage percentage.
Labor Cost, by categories; estimated and actual.
Other Costs, including training, tools, support and hardware; estimated and actual.
Cost Versus Value.
Effort, staff hours by project and phase; estimated and actual (excluding sick, vacation, comp, holidays, personal and
non-project specific time, but including all overtime worked).
Staffing Level, headcount monthly and peak staffing periods.
Defects, discovered defects and their severity, state-at-delivery, phase when detected and phase when introduced.
Productivity Rates.
Estimating Accuracy.
Reuse, modules and artifacts reused.
Process Compliance.
Current Project SEI Level Rating, date of SEI assessment.
Project Success Factors, obstacles to success.
Customer Satisfaction.
Project Characteristics.
Canceled Projects.
Changes in Scope.

Pada tahap perencanaan ini diperlukan juga validasi terhadap kebijaksanaan, standard dan hal-hal lainnya yang terkait dengan keamanan dan selalu dikonfirmasikan ke *stakeholders*-nya (sponsor, users, customers) sampai mereka menyetujuinya.

Ada 8 prinsip untuk dapat mengembangkan piranti lunak secara aman [1] yakni:

1. *Economy of mechanism*: Lakukanlah desain sederhana dan sekecil mungkin
2. *Fail-safe defaults*: Hak akses diberikan sesuai dengan permisi daripada tanpa ada pengecualian
3. *Complete me*diation: Setiap akses dan obyek harus dicek otoritasnya
4. *Open design*: Desain sebaiknya tidak dirahasiakan
5. *Separation of privilege*: Bila memungkinkan, mekanisme proteksi memerlukan paling tidak 2 kunci untuk membukanya, hal ini akan lebih aman daripada hanya mempunyai satu kunci

6. *Least privilege*: Setiap program dan setiap pemakai dari sistem sebaiknya mengoperasikan dengan menggunakan minimal hak akses yang dibutuhkan untuk menyelesaikan pekerjaannya
7. *Least common mechanism*: Meminimalkan jumlah mekanisme umum untuk para pemakai
8. *Psychological acceptability*: Hal ini penting bahwa *human interface* didesain untuk kemudahan penggunaan sehingga para pemakai secara rutin dan otomatis dapat mengaplikasikan mekanisme keamanan secara benar.

2.3 Sektor UKM

Para Software Developer di sektor UKM ini disarankan untuk mengikuti semaksimal mungkin hal-hal tersebut diatas ke stakeholders yang terkait. Hal ini juga terkait dengan sumber daya yang ada dan juga metodologi yang dipakai. Pada umumnya mereka (UKM) sering menggunakan metodologi *Prototyping* dan *Agile* dibandingkan dengan metodologi terstruktur seperti Waterfall, Spiral, Phase, sehingga fase ini tidak kelihatan secara seksama dibandingkan menggunakan metodologi terstruktur. Walaupun demikian mereka harus melakukannnya minimal untuk memastikan bahwa *requirements*nya benar-benar sesuai dan layak dilanjutkan ke tingkat lebih lanjut.

Untuk alasan tertentu, terutama bagi UKM yang memiliki persaingan sangat ketat dengan beban pengembalian modal yang relatif berat sementara margin keuntungan tidak terlalu lebar, issue keamanan system harus menjadi perhatian. Hal ini demi menjaga kelangsungan dari UKM tersebut.

Bila mengambil konsep 8 prinsip di atas maka serangan intelejen marketing lebih dapat dihindari. Berikut ini akan uraikan mengenai 8 prinsip tersebut:

2.1 *Economy of Mechanism*

Piranti lunak dikembangkan secara bertahap, modul-per-modul, segment-per-segment, menurut skala prioritas dan ketersediaan dana. Rancangan perangkat lunak disusun sesederhana mungkin dengan alasan agar cukup dalam pembiayaan pembangunan perangat lunak tersebut, sehingga kebutuhan desain pada bidang lain menjadi dapat dipenuhi.

Hal-hal yang kurang perlu didesain atau merupakan suatu yang umum cukup disdeskripsikan saja, sehingga lingkup dari desain menjadi lebih kecil.

2.2 Fail-safe Defaults

Setiap user ingin mengakses bagian lain yang sebenarnya mereka tidak berkepentingan. Bahkan para eksekutif dan top manajemen menginginkan data transaksional sehingga mereka minta dibuatkan akses sampai level detail.
Jika dibiarkan begitu, maka fungsi control dari seorang admin akan menjadi sangat berat. Bahkan bisa frustasi dan akhirnya seolah-olah semua menjadi admin terhadap user. Terdapat dua kerugian jika suatu data dapat diakses oleh orang lain. Pertama orang tersebut dapat mengambil informasinya untuk keuntungan bukan perusahaan. Kedua orang tersebut dapat melakukan

perubahan data (jika memiliki premisi write) yang berakibat data menjadi tidak valid, tidak konsisten bahkan tidak integritas lagi.

Sistem secara keseluruhan akan ditinggalkan karena tidak akan ada kepercayaan lagi terhadap sistem.

2.3 *Complete mediation*

Pada fase requirement ini, perlu dipetakan dengan jelas, peran dan tanggung jawab dari para user, termasuk mengenai otoritasnya. Otoritas dari user pada umumnya disesuaikan dengan struktural dan job deskripsi yang berlangsung pada perusahaan tersebut.
Bahkan admin pun perlu dicontrol otoritasnya meskipun seorang admin memiliki otoritas yang tinggi.

Setiap object data, object modul, object menu dijabarkan tujuan dan target yang diinginkan sehingga user di UKM dapat terpetakan sesuai fungsi dan tanggung jawabnya. Sebenarnya poin ini hampir sama dengan poin sebelumnya yaitu fail-safe default.

2.4 *Open Design*

Pembangunan perngkat lunak tentu akan berlanjut dari satu siklus ke siklus berikutnya samapi menjadi matang dan dapat memenuhi hampir semua kebutuhan user. Agar dapat berkesinambungan dan informasinya saat ini dapat menjadi pijakan bagi pengembangan yang akan datang, maka desain harus dilakukan secara terbuka.

Jika desain dilakukan secara tertutup, maka tentu akan mengeluarkan biaya yang sangat tinggi bagi pembuatan desain pada tahap berikutnya. Bahkan tidak hanya biaya, tapi juga waktu.

Selain itu kegunaan open desain yaitu sistem dapat dengan mudah terkontrol bagian-bagian mana saja yang memiliki kekurangan, bagian mana saja yang dapat menimbulkan kekacauan, sehingga user dan pengembang sama-sama mnejdai transparan terhadap desain dan rencana kebutuhannya.

2.5 *Separation of privilege*

Untuk poin ini memang sangat tidak tepat bagi UKM yang kurang memiliki kepentingan dan tingkat rahasia yang tinggi dari data. Tapi jika cost tidak terlalu besar untuk implementasinya, prinsip ini sangat baik juga diimplementasikan.

Sederhananya jika dana terbatas, dapat diimplementasikan dari prioritas modul aplikasi mana yang penting diproteksi. Jika dua-duanya sama penting, maka yang menjadi prioritas yaitu yang memiliki frekuensi akses tertinggi. Hal ini hanya untuk menunjukan bahwa proteksi keamanan data pada fase requiremen dan analisa perlu dipertimbangkan.

Dua kunci ibaratnya jika suatu perusahaan memiliki rekening di suatu bank. Dimana jika akan dilakukan pengambilan uang, maka aplikasi pengambilan uang harus ditandatangani minimal dua pihak yang berwenang, misalkan kepala bagian keuangan dan direktur. Atau dua kunci yang berbeda dapat digambarkan juga sebagai proteksi ganda, dimana setelah user memasukan kata kunci dan berhasil maka user diminta memasukan kata kunci lain yang lebih secure. Misalkan

seteleha masukan user dan memasukan password dan berhasil, maka user harus memasukan kata kunci yang lain yang telah didefinisikan, atau memasukan angka/ pin tertentu.

2.6 *Least privilege*

Prinsip ini merupakan sambungan dari prinsip sebelumnya mengani otoritas dan hak akses. Issue yang penting dari prinsip ini yaitu seorang user yang memiliki hak akses lebih, menggunakan otoritasnya secara maksimal dalam menyelesaikan pekerjaannya . Ini mengakibatkan pemborosoan biaya dan waktu serta memungkinkan membahayakan keamanan sistem. Ilustrasi sederhananya yaitu, untuk membaca suatu file informasi umum dibuka dengan menggunakan otoritas Administrator atau root. Yang sebenarnya administrator atau root lebih tepat untuk membuka atau memperbaiki sistem server atau aplikasi.

2.7 *Least common mechanism*

Mekanisme umum perlu diminimalkan agar UKM memiliki nilai keuntungan kompetitif dibandingkan yang lain. Namun demikian mekanisme tersebut tetap harus memiliki desain sederhana dan terbuka.

Dari sisi keamanan, mekanisme umum mudah terbaca bagi kompetitior, bahkan lebih parahnya dapat diprediksi sehingga hal ini jelas tidak menguntungkan bagi UKM.

2.8 *Psychological Acceptability*

Sebagian orang berpendapat bahwa dengan user interface yang rumit memungkinkan orang lain akan menghindari penggunaannya sehingga akan menyebabkan data atau sistem lebih aman. Padahal ini juga akan membawa dampak atau pengaruh yang kurang baik bagi user internal maupun adminnya.

Salah satu akibatnya bahwa user itu tidak digunakan maksimal sehingga tujuan pembuatan perangkat lunak tersebut menjadi sia-sia.

Kenyataannya pembuatan desain interface akan memperbaiki kehandalan dari sistem. Selain user internal menjadi lebih familiar juga akan terjaga dari kesalahan entry atau update data yang akan mengakibatkan fatal bagi data dan sistem bersangkutan maupun keseluruhan. Biaya training pun akan dapat ditekan dan dialihkan untuk pengembangan yang lainnya.

Pihak luarpun akan mudah menggunakannya namun terproteksi dengan baik. Dan piohak admin akan dapat memonitor dengan mudah.

III. KEAMANAN PADA FASE ANALISA *DESIGN*

Fase Analisa ini adalah untuk menginvestigasi lebih rinci sistem yang saat ini sedang digunakan seperti model proses, data dan informasi; kesempatan untuk mengidentifikasi area yang perlu diimprovisasi, dan mengembangkan konsep baru untuk sistem yang akan dikembangkan menjadikan lebih baik.

Biasanya akan muncul pertanyaan-pertanyaan sebagai berikut dalam fase analisa ini [2]:

- Why do the problems exits?
- Why were certain methods of work adopted?
- Are there alternative methods?
- What are the likely growth rates of data?

Dengan kata lain pertanyaan-pertanyaan tersebut atas sebagai usaha untuk memahami semua aspek dari sistem yang saat ini sedang berjalan dan mengapa sistem yang sedang berjalan ini harus dikembangkan serta diperlukan suatu perubahan yang lebih baik sehingga akan memperoleh manfaat yang besar pada penerapan sistem informasi yang diusulkan.

Fase Perancangan ini untuk memutuskan perancangn sistem yang diusulkan baik model konseptual (seperti arsitektur design), model logik (seperti model data & proses) dan model fisik (seperti struktur data & program, user interface, laporan, dan bagaimana sistem akan dioperasikan menggunakan teknologi tertentu seperti kebutuhan piranti keras, piranti lunak, dan arsitektur jaringan).

3.1 Hal-hal yang terkait pasa Fase Perancangan/Desain

Pada fase Desain ini, hal-hal yang diperhatikan antara lain adalah:

- **Bagaimana mengidentifikasi dan mengevaluasi ancaman**

  Menggunakan threat modeling untuk secara sistematis mengidentifikasi ancaman jauh lebih baik dari pada menerapkan keamanan secara sembrono. Pemodelan bisa dilakukan dengan memberi peringkat terhadap ancaman yang mungkin terjadi berdasarkan resiko serangan, frekuensi kejadian, yang dikaitkan dengan kurugian atau kerusakan potensial yang bisa terjadi. Hal ini bisa memudahkan kita dalam menangani ancaman dengan urutan yang sesuai.

- **Bagaimana membuat desain yang aman**

  Sebaiknya menggunakan prinsip-prinsip desain yang sudah dicoba dan teruji dan fokus pada area kritis dimana butuh pendekatan yang benar dan sering terjadi kesalahan yang meliputi: validasi input, autentifikasi, otorisasi, manajemen konfigurasi, proteksi data sensitif, manajemen sesi, kriptografi, manipulasi parameter, manajemen eksepsi, dan pertimbangan audit dan logging. Harus pula memberi perhatian yang serius pada isu-isu deployment termasuk topologi, infrastruktur jaringam, kebijakan keamanan, dan prosedur.

- **Bagaimana memeriksa arsitektur dan desain**

  Desain aplikasi harus diperiksa dalam hubungannya dengan lingkungan deployment dan kebijakan kemanan yang terkait. Perlu membuat batasan yang ditentukan berdasarkan kemanan pada lapisan insfrastruktur termasuk jarimangan perimater, firewall, remote application servers, dan sebagainya. Sebaiknya juga menggunakan kategori vulnewability untuk membagi aplikasi, dan menganalisa pendekatan yang diambil dalam tiap area.

Untuk acuan keamanan pada fase ini bisa melihat pada *security check-list* di lampiran.

3.2 Sektor UKM

Keterbatasan sumberdaya manusia biasanya merupakan kendala dan juga metodologi yang dipakai, untuk itu perlu dilakukan “Joint Application Development” dan jiga pemahaman bersama pentingnya keamanan dalam pengembangan sistem informasi serta adanya pelatihan.

Untuk mengidentifikasi dan mengevaluasi ancaman, harus memiliki pengetahuan yang lebih. Sementara sumber daya di UKM yang tersedia tidak memenuhi kebutuhan seperti waktu, biaya, dan skill.

Melalui JAD, akan terbentuk pemahaman dan pengetahuan baru mengenai proses perancangan aplikasi yang aman. Dengan JAD juga, aktivitas setiap langkah akan ikut termonitor dan terkendali. Sehingga antara pengembang perangkat lunak dan personal-in-cgharge di UKM dapat berkomunikasi secara transparan, dan menghilangkan kemungkinan kecurangan, saling mengidentifikasi lubang-lubang terjadinya kelemahan sistem.

JAD membantu dalam pendanaan training, peningkatan skill, dan penghematan waktu implementasi. Namun tentunya staf yang ditunjuk untuk mengikuti JAD harus minimal memiliki penegtahuan dasar sehingga tidak menjadi beban bagi pengembang.

Perancangan perangkat lunak yang menginginkan terjamin keamanannya akan dapat dihasilkan lebih maksimal. Identifikasi akan lebih teliti dan potensi kemungkinan ancaman akan bisa diantisipasi. Rancangan yang aman dan arsitektur yang baik akan meningkatkan performansi sistem. Pada siklus pengembangan berikutnya akan lebih cepat menuju maturity.

IV. KEAMANAN PADA FASE PENGEMBANGAN

Fase Pengembangan sebenarnya masuk fase implementasi kalau dilihat dari fase-fase yang ada dalam SDLC [2], tapi sehubungan topik ini berkaitan dengan sistem keamanan yang perlu pembahasan khusus maka kami menuliskannya sendiri terpisah dari fase implementasi.

4.1 Pemrograman dan Percobaan

Pada fase pengembangan ada 2 topik besar yakni pemrograman dan percobaaan (testing), adapun secara umum hal-hal yang perlu diperhatikan dapat diuraikan sebagai berikut:

- **Bagaimana menulis kode yang aman**

Gunakan nama yang kuat untuk menandai secara digital pekerjaan kita. Kurangi profil serangan dengan mengikuti prinsip-prinsip desain object oriented, lalu gunakan keamanan akses kode untuk membatasi kode mana bisa memanggil kode lainnya. Gunakan penanganan eksespi yang terstruktur untuk menjaga informasi yang sensitif agar tidak keluar dari batas yang terpercaya dan untuk mengembangkan kode yang lebih kuat.

· **Bagaimana menangani eksespi yang aman**

Jangan pernah mengungkapkan informasi tentang sistem internal atau aplikasi seperti stakj traces, SQL statement fragments, dan sebagainya. Pastikan informasi jenis ini tidak sampai pada end user atau keluar dari batas yang terpercaya.

Jangan mencatat data yang sensitif atau pribadi seperti password. Ketika mencatat laporan eksespi, jika input dari user dimasukkan dalam pesan yang dicatat, validasilah atau bersihkan terlebih dahulu, misalnya jika pesannya dalam format HTML, maka harus di encode terlebih dahulu untuk menghindari injeksi script.

· **Bagaimana melaksakan pemeriksaan keamanan pada managed code**

Gunakan alat analisa seperti FxCop untuk menganalisa file binary dan untuk memastikan sesuai dengan design guideline dari framework yang dipakai. Perbaiki vulnerability yang dapat ditemukan oleh alat tersebut.Gunakan fasilitas pencarian text untuk scan kode sumber. Berikan perhatian lain pada vulnerability akibat SQL injection dan cross-site scripting.

· **Bagaimana mengamankan alat kerja pengembang**

Tempat kerja harus juga mendapat perhatian dalam masalah keamanan. Pengembang harus mengamankan alat kerjanya seperti: account, protokol, port, service, share, file dan direktory, dan registry. Hal penting lainnya adalah memastikan bahwa workstation selalu mengikuti update dan patch terbaru.

· **Bagaimana menulis least privileged code**

Kita dapat membatasi kode apa yang bisa dijalankan berdasarkan account. Untuk itu dapat digunakan code access security untuk membatasi resources dan operasi yang diperbolehkan dengan membuat kebijakan terlebih dahulu. Jika kode tidak membutuhkan mengakses sebuah resource atau operasi, maka dapat digunakan declarative security attributes untuk memastikan kode tersebut tida diberi hak oleh administrator. Intinya, berikan akses pada yang membutuhkan saja.

· **Bagaimana membatasi file I/O**

Penggunaan code access security dapat dilakukan untuk membatasu kemampuan kode dalam mengakses area dari sistem file dan menjalankan file I/O. Sebagai contoh, sebuah aplikasi web dapat dibatasi hanya bisa mengakses file pada virtual directory yang dimilikinya.

· **Bagaimana mencegah SQL injection**

Untuk mengakses data, sebaiknya digunakan stored procedures yang mempunyai parameter. Penggunaan parameter tersebut untuk memastikan nilai input sudah dicek tipe dan panjangnya. Parameter juga diberlakukan untuk menjamin sebagai nilai yang aman dan bukan kode yang dapat dieksekusi dalam database. Jika tidak dapat menggunakan stored procedures, sebaiknya menggunakan SQL statement dengan parameter. Jangan pernah membangun SQL statement dengan langsung memasukkan nilai input dalam SQL command. Pastikan aplikasi memberikan hak akses ke database seperlunya saja.

· **Bagaimana mencegah cross-site scripting**

Untuk mencegah cross-site sripting, validasi semua input termasuk tipe, panjang, format, dan range dan yang penting encode outputnya. Sebagai contoh, encode form fileds, query string parameters, cookies, dan sebagainya.

· **Bagaimana mengatur rahasia**

Carilah pendekatan alternatif guna menghidari menyimpan informasi rahasia di urutan pertama. Jika harus menyimpan informasi tersebut, jangan pernah menyimpannya dalam text biasa di kode sumber atau di file konfigurasi. Jangan lupa enkripsi informasi rahasia dengan Data Protection Application Programming Interface (DPAPI) untuk menghindari isu manajemen key.

· **Bagaimana memanggil unmanaged code secara aman**

Berilah perhatian tersendiri pada paramater yang memasuki atau datang dari unmanaged API, dan hati-hati dengan kemungkinan terjadinya buffer overflows. Validasi panjang dari parameter string input dan output, cek array, dan hati-hati dengan panjang file path. Gunakan permintaan izin tertentu untuk menjaga akses ke unmanaged resources sebelum menyatakan izin unmanaged code. Gunakan peringatan untuk meningkatkan performance.

· **Bagaimana melakukan validasi input yang aman**

Batasi, tolak, bersihkan input merupakancara yang lebih mudah dalam melakukan validasi data untuk tipe, pola, range yang valid dan sudah diketahui terlebih dahulu kemudian mencari karakter yang jelek. Untuk input berupa string bisa digunakan ekspresi regular.

· **Bagaiaman mengamankan Forms authentication**

Harus ada pembagian area mana yang bisa diakses oleh anonim dan area mana yang hanya bisa diakses oleh user dengan menggunakan otentifikasi. Jika aplikasi berbasis web, gunakanlah Secure Sockets Layer (SSL). Batasi juga jangka waktu sesi dan pastikan authentication cookie hanya melalui HTTPS saja. Enkripsi authentication cookie dan jangan gunakan untuk tujuan personalisasi. Untuk personalisai, gunakan cookie terpisah selain authentication cookie.

Untuk acuan keamanan pada fase ini bisa melihat pada *security check-list* di lampiran.

4.2 Sektor UKM

Mirip dengan fase sebelumnya keterbatasan sumberdaya manusia dan metodologi juga masih merupakan kendala sehingga mungkin sulit untuk melakukan tahapan-tahapan yang diperlukan dalam pemrograman dan ujicoba sehingga menghasilkan aplikasi yang aman diantaranya:

-Pemahaman

Perlu adanya *training*

-Test

Diperlukan Unit Test, Integration test, System test dan Acceptance Test. Kendala SDM bisa diatasi dengan JAD dan menggunakan siswa/mahasiswa untuk magang sehingga hasilnya akan lebih optimal

Hal yang mendasar perbedaan UKM dengan corporate yaitu dari kekuatan modal. Apalagi UKM yang tidak berfokus pada pengelolaan IT dan pemanfaatannya lebih minim, tentu alokasi dana investasinya sangat minim. Sementara kebutuhan dalam pemrograman dan testing untuk menghasilkan standar yang baik harus tetap dilakukan.

Memang tidak semua poin-poin di atas harus dipenuhi secara maksimal. Ada tingkat toleransi dan standar minimal yang dapat dipenuhi. Tingkat kedalaman disesuaikan dengan karakteristik dari aplikasi tersebut.

Sebagai contoh untuk menjawab “How to secure input validation” tentunya akan menjadi prioritas jika aplikasi menuntut input user harus beanr-benar valid, sesuai standar, dan menjadi penting bagi aplikasi dan sistem tersebut. Begitu pula untuk menjawab pertanyaan "How to prevent SQL injection?". Pertanyaan ini akan menjadi prioritas jika didalamnya terdapat konten-konten informasi yang sangat penting dan dipublish ke masyarakat sebagai gambaran citra dari perusahaan.

Pada prinsipnya, UKM harus memiliki strategi untuk menentukan mana yang akan dilakukan terlebih dahulu dari pertanyaan-pertanyaan yang harus dijawab di atas, dengan disesuaikan dana yang tersedia.

Semakin banyak dananya, maka object dan pertanyaan akan semakin banyak pula dijawab. Dan jika sudah bisa dijawab, maka selanjutnya pada siklus berikutnya akan dijawab kembali dengan solusi yang lebih detil dan baik.

V. KEAMANAN PADA FASE IMPLEMENTASI

Fase terakhir dari tahapan SDLC ini adalah fase Implementasi yang terdiri atas 3 tahap [1] yakni:

- Tahap Konstruksi/Pemrograman dan Testing

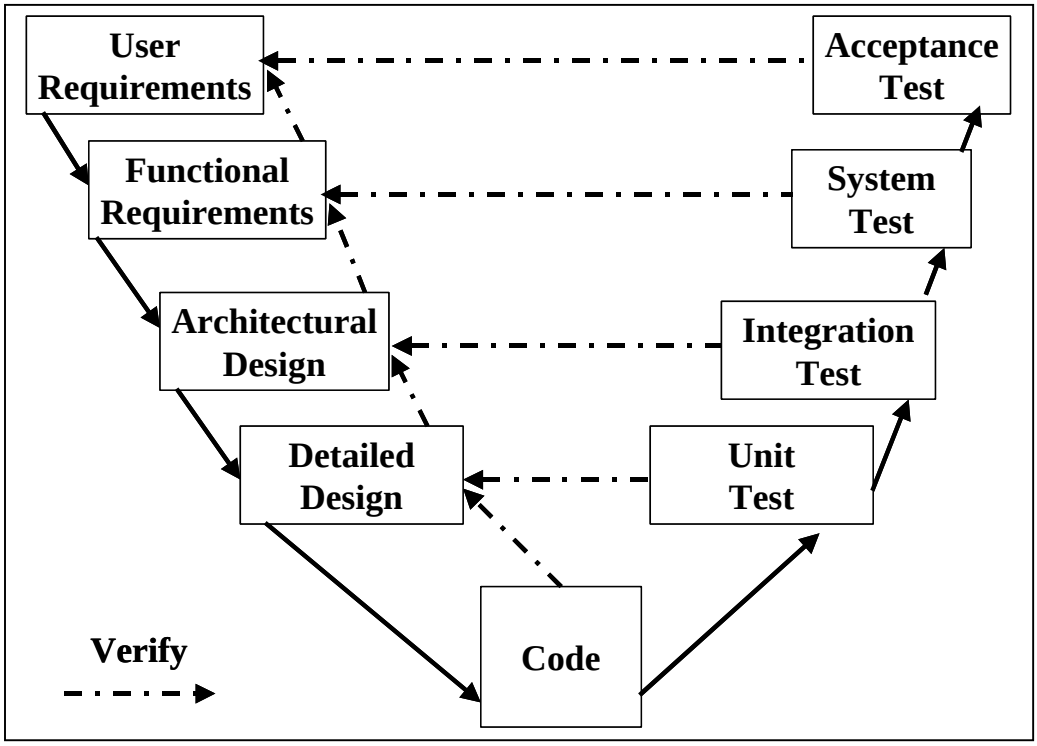
Tahapan ini merupakan bagian dimana sistem secara riil harus segera dimiliki dengan beberapa alternatif diantaranya melakukan konstruksi/pemrograman sendiri, *outsourcing*, atau membeli paket aplikasi yang sudah ada dipasaran. Biasanya tahapan ini menjadikan hal yang mendapatkan perhatian paling besar dikarenakan tahapan ini dianggap membutuhkan biaya yang paling banyak dengan waktu yang relatif paling lama dari tahapan pengembangan sistem informasi ini.

Selain itu juga diperlukan tahap percobaan (*testing*) sistem informasi tersebut untuk menyakinkan semua tahapan yang dilakukan sampai pemrograman dan juga verfikasi serta validasi sesuai yang diilustrasikan pada gambar 5.

Bagian telah dijelaskan pada bab 4 – Fase pengembangan

5.1 Tahap Penerapan/Instalasi

Tahapan ini dilakukan setelah tahap ujicoba telah dilalui dan para pemakai telah menyetujuianya sehingga sistem informasi yang baru dibangun ini akan dapat segera menggantikan sistem yang saat ini sedang berjalan. Tahapan ini bisa dilakuan per modul yang sesuai dengan kebutuhannya atau sekaligus menggantikannya baik dengan cara parallel run atau dengan langsung cut over dan tentunya setelah diadakan sistem pelatihan baik untuk para pemakai maupun untuk sistem administrasi, operasional dan dukungan teknis level pertama (helpdesk).



Gambar 5 Verifikasi dan Validasi dalam Pengembangan Sistem Informasi

5.2 Tahap Pemeliharaan & Review

Tahapan ini termasuk tahapan yang selama sistem informasi sedang digunakan (implementasi) sehingga dibutuhkan suatu dukungan teknis dan pemeliharaan sehingga kalau ditemukan bugs yang belum terdeteksi pada saat ujicoba, maka developers bisa segera memperbaikinya. Kemudian bila ada perubahan dan atau usulan-usulan baru yang diinginkan baik para pemakai maupun manajemen, maka perlu adanya team review atau disebut Change Control Board untuk mengevaluasinya sehingga dimungkinkan untuk dimasukkan sebagai rencana untuk melakukan perubahan di versi selanjutnya.

Hal-hal perlu diperhatikan dalam fase ini diantaranya adalah:

- **Bagaimana mengimplementasikan patch management**

Jalankan update patch secara berkala untuk memastikan mengikuti update dan patch terbaru. Jika menggunaka produk microsoft, dapat menggunakan Microsoft Baseline Security Analyzer (MBSA) untuk mendeteksi adanya patch terbaru. Jangan lupa melakukan backup server sebelum memasang patch, uji coca terlebih dahulu sebelum install ke server produksi. Juga, ikuti milis atau notifikasi lain dari vendor yang ada.

- **Bagaimana mengamankan database server**

Gunakan metodologi umum untuk evaluasi account, protokol, service, share, file dan direktori, serta registry. Evaluasi juga pendekatan otorisasi yang digunakan dalam login, user, dan role. Jangan lupa untuk selalu mengikuti perkembangan update dan patch dari server database yang digunakan.

- **Bagaimana mengamankan application server**

Evaluasilah account, protokol, service, share, file dan direktori, serta registry. Gunakan Internet Protocol Security (IPSec) atau SSL untuk mengamankan saluran komunikasi antara webserver dan application server, antara application server dan database server. Periksa keamanan dari Enterprise Services applications, Web services, dan remoting applications. Batasi port mana saja yang boleh dipakai oleh client untuk tersambung ke application server.

- **How to secure Web services**

In cross-platform scenarios and where you do not control both endpoints, use the Web Services Enhancements 1.0 for Microsoft .NET (WSE) to implement message level security solutions that conform to the emerging WS-Security standard. Pass authentication tokens in Simple Object Access Protocol (SOAP) headers. Use XML encryption to ensure that sensitive data remains private. Use digital signatures for message integrity. Within the enterprise where you control both endpoints, you can use the authentication, authorization, and secure communication features provided by the operating system and IIS.

- **How to secure Enterprise Services**

Configure server applications to run using least privileged accounts. Enable COM+ role-based security, and enforce component-level access checks. At the minimum, use call-level authentication to prevent anonymous access. To secure the traffic passed to remote serviced components, use IPSec encrypted channels or use remote procedure call (RPC) encryption. Restrict the range of ports that Distributed COM (DCOM) dynamically allocates or use static endpoint mapping to limit the port range to specific ports. Regularly monitor for Quick Fix Engineer (QFE) updates to the COM+ runtime.

- **How to secure session state**

You need to protect session state while in transit across the network and while in the state store. If you use a remote state store, secure the communication channel to the state store

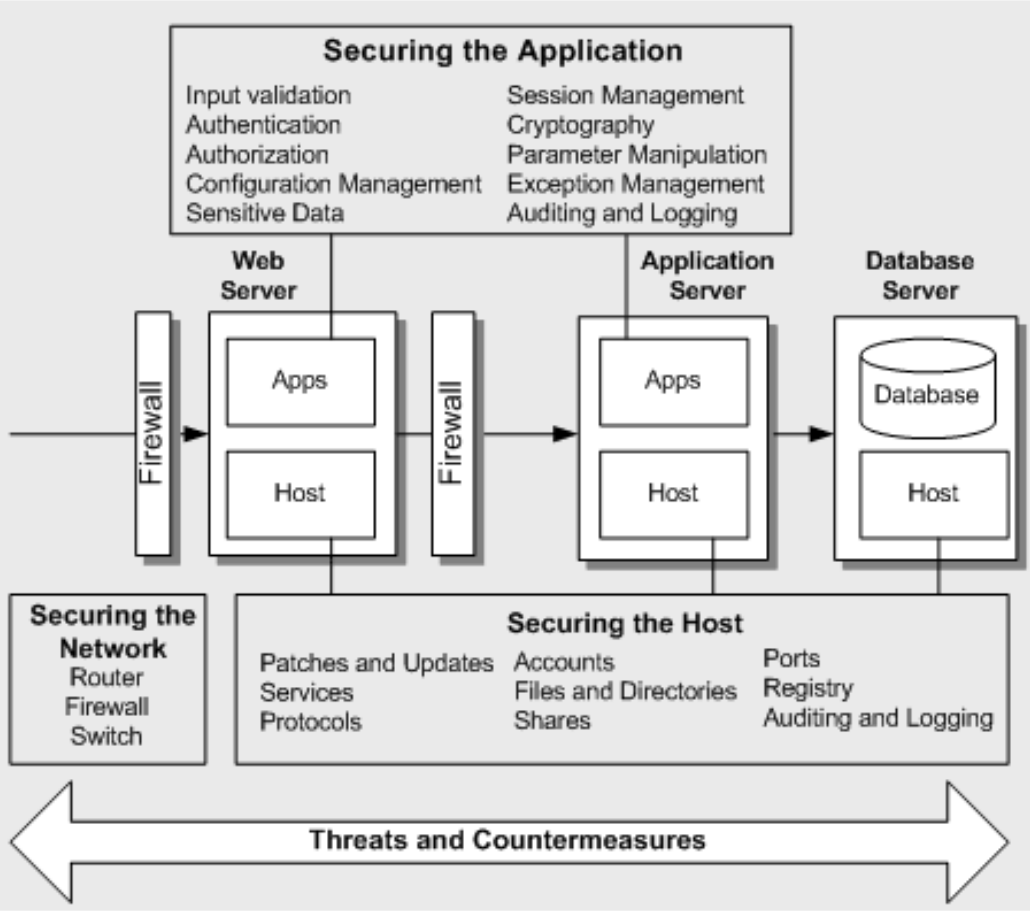
using SSL or IPSec. Also encrypt the connection string in Machine.config. If you use a SQL Server state store, use Windows authentication when you connect to the state store, and limit the application login in the database. If you use the ASP.NET state service, use a least privileged account to run the service, and consider changing the default port that the service listens to. If you do not need the state service, disable it.

- **Bagaimana mengatur konfigurasi aplikasi secara aman**

Administrasi secara remote harus dibatasi atau bahkan dihindari. Otentifikasi yang kuat harus ada untuk interface administrasi. Batasi juga akses ke informasi konfigurasi yang tersimpan.

- **Bagaimana mengamankan dari serangan denial of service**

Pastikan konfigurasi TCP/IP stack di server cukup kuat untuk menjaga dari serangan seperti SYN floods. Buat konfigurasi pada server dengan membatasi permintaan yang aman seperti POST dan GET serta hindari permintaan speri PUT. Pasang firewall untuk mengidentifikasi request yang berulang dalam waktu pendek atau yang mengirim paket data dengan ukuran besar.



Gambar 6 Komposis sistem keamanan pada aplikasi *enterprise*

- **Bagaimana membatasi file I/O**

Kita dapat membuat kebijakan code access security untuk memastikan sebuah perintah atau seluruh aplikasi dibatasi pada hak akses terhadap sistem file tertentu. Kebijakan tersebut juga bisa dibuat untuk menentukan bagaimana cara menkases sistem file tertentu.

- **Bagaimana melakukan remote administration**

Untuk melakukan administrasi secara remote, sebaiknya menggunakan saluran yang terenkripsi dan membatasi komputer mana yang dapat digunakan untuk melakukan adminstrasi ke server secara remote. Batasan yang bisa dibuat misalnya alamat IP, alamat MAC, user account, dsb.

Pada fase implementasi / operasional ini dibutuhkan sebuah team sebagai *Change Control Board* untuk menjembatani antara kebutuhan pemakai dengan *software developers* sehingga kalau ada perubahan-perubahan yang diusulkan oleh para pengguna harus dievaluasi beberapa aspek diantaranya factor keamanannya seperti *integrity issue* dan hal lainnya.

5.3 Sektor UKM

Mirip dengan fase sebelumnya keterbatasan sumberdaya manusia dan kompleksitas dalam implementasi apalagi operasional diperlukan kerja sama dengan bagian terkait.

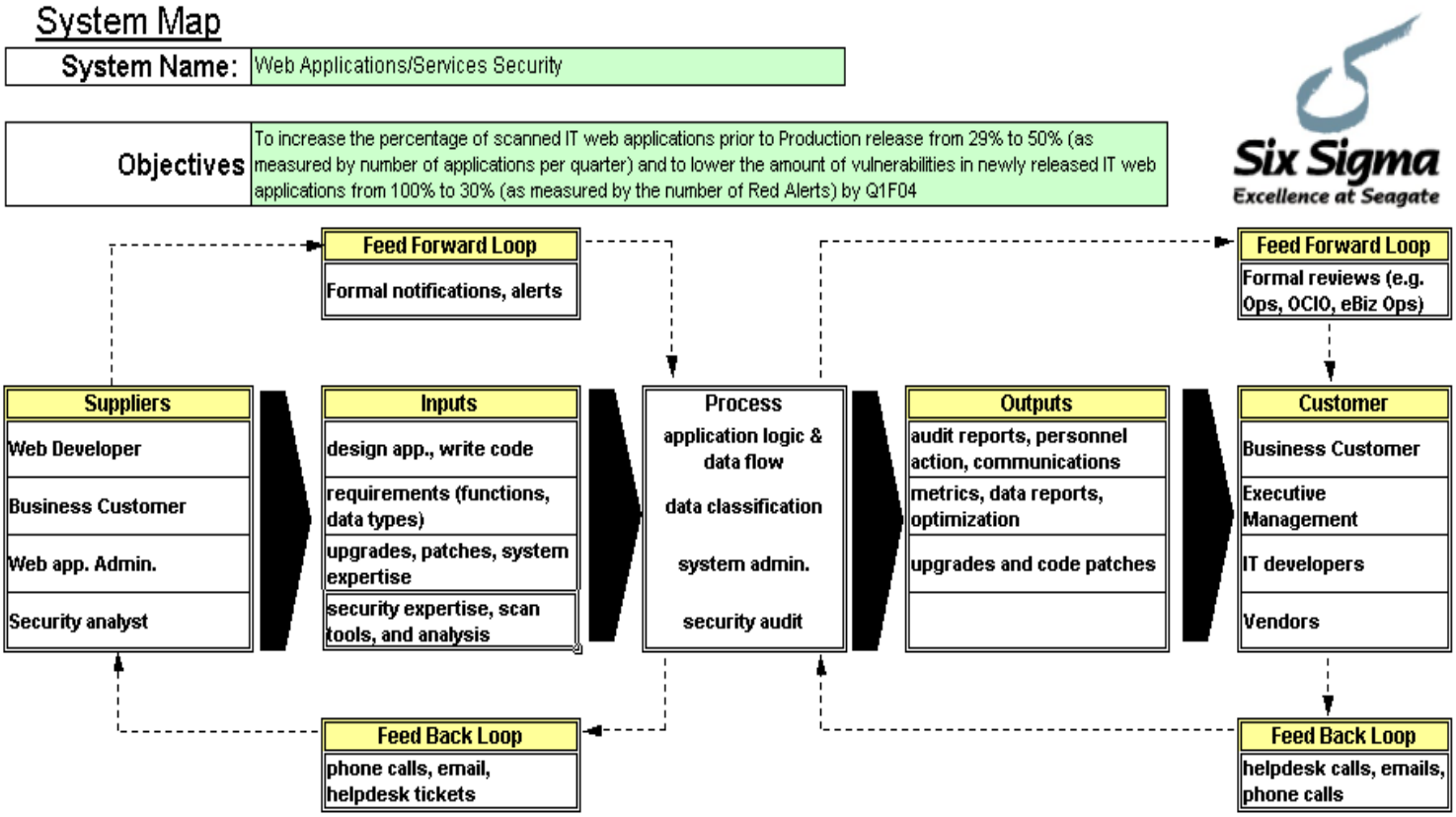
Strategi yang diterapkan juga akan mirip dengan fase sebelumnya dimana akan digunakan metode skala prioritas dalam menjawab pertanyaan-pertanyaan dalam fase implementasi dan operasional ini. Inti dari strategi ini yaitu bagaimana tercipta implementasi dan pemeliharaan sistem dengan cara yang mudah, biaya yang murah dan kualitas operasional dan sistem tetap terjamin.

Perlu adanya upaya untuk meningkatkan kualitas walaupun sumber dayanya terbatas.

VI. PERANGKAT UNTUK PENINGKATKAN KUALITAS

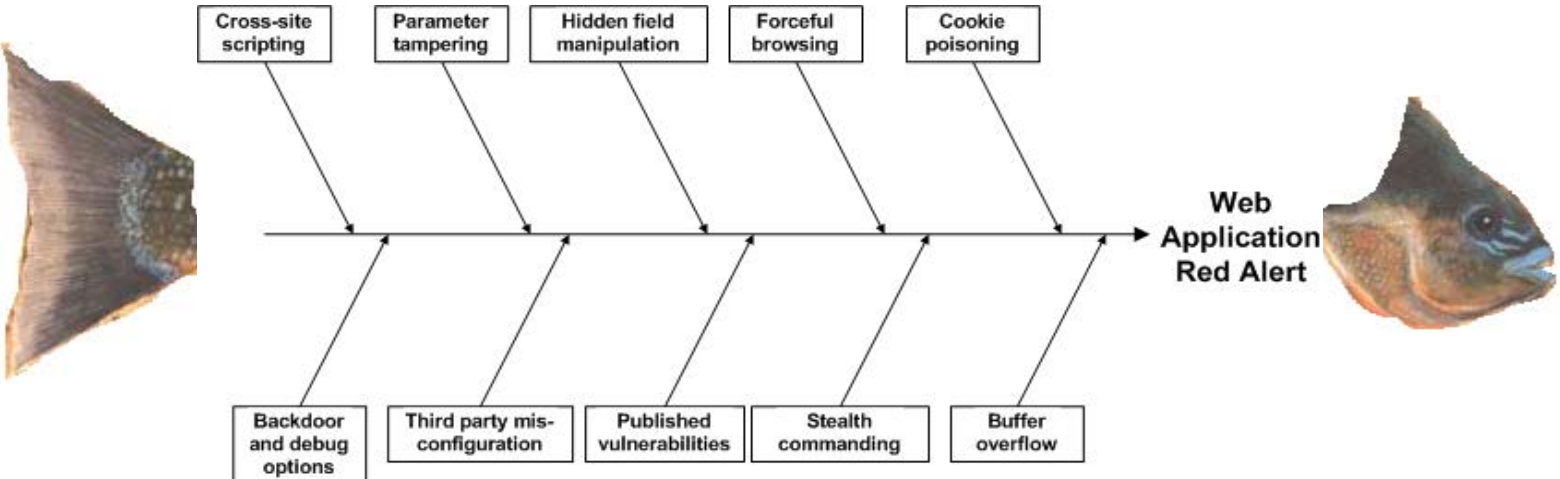
Seperti yang telah dituliskan sebelumnya bahwa belum ada QA tool yang dapat mendeteksi factor keamanan dari system informasi/aplikasi yang dibuat. Berikut ini ada 3 tools yang dapat membantu unuk

6.1 Six Sigma



Source: An Evolution of Putting Security into SDLC [1]

6.2 *Diagram Fishbone*



Source: An Evolution of Putting Security into SDLC [1]

6.3 *Cause Effect*

		1	2	3	4	5	6	7	8	9	10
	EFFECTS	Cookie poisoning	Forceful browsing	Hidden field manipulation	Parameter tampering	Cross-site scripting	Buffer overflow	Stealth commanding	Published vulnerabilities	Third party misconfiguration	Backdoor and debug options
CAUSES		Cookies can be modified by client	Application do not force a browsing order on client	Hidden fields used to track session	URL parameters are changed	Forms accept metatags	More data than the application expects	Form fields accept upload of malicious code	"Bugs" and security holes in 3rd party code	Insecure default settings in 3rd party application	Developers insert backdoors into applications and forget to remove for production
		Return of unarthorized information to application	Hacker can jump directly to pages normally controlled by authentication mechanisms	Hidden fields can be seen using View Source	Null value causes application to enter undefined state	Metatag characters are not filtered by the application	Incoming data size is not checked	Site defacement or server execution of uploaded code	Patches are delayed while hackers have published exploit code	Unclear or lack of configuration procedures	Debugging is turned on
		Previously saved cookies are modified and sent as current cookies to server		Client data is not validated by server	Parameters are not checked by application			Field validation on server are not checked	Patches are not keep updated	Mis-configurations are published on hacker sites	Backdoors do not require passwords
		Cookier are not encrypted						Database statements (insert, delete) are not validated			Error messages and comments in the code reveal vulnerabilities

Source: An Evolution of Putting Security into SDLC [1]

VII. PENUTUP

7.1 Kesimpulan

- Sistem keamanan terutama pada pengembangan Sistem Informasi/Aplikasi banyak ditujukan untuk perusahaan-perusahaan besar hal ini dikarenakan karena faktor ancaman nilai eknomis/bisnis yang jauh lebih besar terjadi dibandingkan sektor UKM
- Ada 10 jenis kerawanan yang sewaktu-waktu bisa berubah dan juga ada 10 prinsip system keamanan yang sebaiknya sector UKM mengimplementasikannya seoptimal mungkin
- Tiap fase dalam SDLC ada hal-hal penting yang berkaitan dengan sistem keamanan dari mulai perencanaan, analisa & design, pengembangan dan implementasi.

7.2 Saran

- Tulisan ini jauh dari sempurna, oleh karena itu saran dan perbaikan sangat berguna bagi tulisan ini
- Bila ada pilihan untuk menambah fitur baru dan ada permasalahan pada sistem keamanan, maka sebaiknya kami memilih untuk menyelesaikan masalah pada system keamanan dari pada menambah fitur baru walaupun dibutuhkan sekali.

DAFTAR PUSTAKA:

[1] Coleman Curtis, Case Study: An Evolution of Putting Security into SDLC, Diambil November 10, 2005 dari **http://www.owasp.org/docroot/owasp/misc/COLEMAN-Putting_Security_IntoSDLC-OWASP_v2.ppt**

[2] Dennis, Alan, Wixom, Barbara dan Tegarden, David, Systems Analysis and Design With UML 2.0 An Object-Oriented Approach,, John Wiley & Sons, Inc Efraim, 2005

[3] Lebanidze Eugene, Securing Enterprise Web Applications at the Source, Diambil November 10, 2005 dari **http://www.owasp.org/docroot/owasp/misc/Securing_Enterprise_Web_Applications_at_the_Source.pdf**

[4] Damianides, Marios dan Parkinson, J.A. Michael, CISA Review Manual 2005, Information Systems Audit and Control Association (ISACA), 2005

[5] Krutz, L. Ronald dan Vines, Dean Russell, The CISSP Prep Guide: Gold Edition, Wiley Publishing, Inc., 2004

[6] Redwine, T. Samuel dan Davis Noopur, Processes To Produce Secure Software, National Cyber Security Summit - USA, 2004

[7] Microsoft: Introduction to Web Application Security. Diambil November 10, 2005, dari **http://www.microsoft.com/security/guidance/topics/default.mspx**

LAMPIRAN

A. *SECURITY CHECK LIST*

Untuk menghasilkan piranti lunak dari suatu sistem informasi yang dikembangkan secara aman perlu adanya petunjuk dan acuan diantaranya adalah *Check List* untuk membantu pihak yang terkait dalam pengembangan piranti lunak seperti arsitek/analis, developer/programmer, maupun system administrasi untuk meverifikasi dan mevalidasi semua hal-hal yang diperlukan dalam menghasilkan piranti lunak yang aman.

Adapun check list tersebut ada dua versi yakni:

1. Versi OWASP (Open Web Application Security Project)

Authentication:	
How do users other entities log into the system?	
What type of information is contained in the system (excluding login Id and password) Is it considered confidential/Highly confidential information/Top Secret?	
What type of users use this system? E.g., External customers, Internal users, "Trusted users" General public... (Please specify)	
Are Pins at least X characters and Passwords at least XX?	Yes :☐ No:☐
Are Passwords of Complex Composition?	Yes :☐ No:☐
Is the PIN or Password stored in a HASH format in the database?	Yes :☐ No:☐
Is the account locked in the DB if there are between 3 and X bad attempts to login to the account?	Yes :☐ No:☐
GET, POST & Encryption: Are you only ever sending the ID & Password via the POST method?	Yes :☐ No:☐

Ensure that GET is not used to send sensitive data as the information is logged in clear text even if SSL is used. SSL only encrypts data in transit – not at the destination point. If POST is used the HTTP body is not logged. However the POST method still sends data as clear text, thus encryption is vital. Ensure that encryption is used for sensitive data at the application level.	
Are you ever passing the user's password in the clear over the network internally?	Yes :☐ No:☐
Are you storing the user's password anywhere other than the database? (In hidden field/cookie/session object?) If yes, where? (Please specify)	Yes :☐ No:☐

Authorization:	
How do you ensure that a given user is only allowed to see data that they have been authorized to view? (*Please specify)*	
How granular is your Authorization check? (*Answer the questions below*)	
Does it support access levels? e.g. user, group, admin, etc.	Yes :☐ No:☐
Does it support permissions per access level? e.g. read, write, delete, create, etc.	Yes :☐ No:☐
Does every page have an explicit check to your authorization logic? A check is made on every request received from the browser.	Yes :☐ No:☐

Data Validation	
Are you validating every source of input from the browser that comes into your application? (*Answer questions below*)	Yes :☐ No:☐
All form data (text boxes, select boxes, hidden fields, etc.)	Yes :☐ No:☐

Cookies used by your application	Yes :☐ No:☐
Headers (Things such as REFERER, USERAGENT, Content-Length, Content-Type etc.)	Yes :☐ No:☐
What type of validation rules does the application use? (*Answer each question below*)	Yes :☐ No:☐
Does the application exploit any Data validation functions contained in the underlying framework? (E.g. Commons Validator .NET Security etc). http://www.owasp.org/software/validation.html	Yes :☐ No:☐
Uses validation rules that are specific to field data	Yes :☐ No:☐
Uses generic validation rule for all fields	Yes :☐ No:☐
Are you using the strongest level of validation possible?	Yes :☐ No:☐
Did you remember to check for maximum length for each field?	Yes :☐ No:☐
If data fails validation for being too long, do you send that information to your logs in its entirety?	Yes :☐ No:☐
Do you encode bad data before echoing it back to the browser to prevent cross-site scripting issues?	Yes :☐ No:☐

Error Handling	
Do you have an Error handling strategy?	Yes :☐ No:☐
Did you build the logic in your application to 'fail securely'? (Basically establishing a default fail stance, and only letting something continue when it has explicitly passed?)	Yes :☐ No:☐

Logging/Debugging/Error Messages	
Are you making sure NOT to log any highly confidential/secret information to your logs, such as password?	Yes :☐ No:☐
Have you made sure that only business error messages (such as 'insufficient funds' or 'you do not have rights to that function') are being sent back to the browser?	Yes :☐ No:☐
Have you ensured that your main error page for serious system problems is generic and not too revealing of the problem that occurred on the server?	Yes :☐ No:☐
Have you made sure that the logging strategy recognises the following: • User input you have logged, possibly as a result of an error, is clean of things that might cause a cross-site scripting attack if later viewed through a browser?	Yes :☐ No:☐

• Logging user input without length checks?	

Cookie/Session Management	
Have you encrypted all confidential or highly confidential information that is being passed in cookies?	Yes :☐ No:☐
If you made your own session identifier, have you made sure that it is random & unique?	Yes :☐ No:☐
Have you made sure that your sessions expire both after disuse (HTTP timeout) as well as after a set length of time (Hardlimit)?	Yes :☐ No:☐
Do you have a session validation strategy? When is the user's session validated (ideally should be done on every request received)? (*please specify)*	Yes :☐ No:☐

2. Detail dan Spesifik (Versi Microsoft)

Deployment and Infrastructure Considerations

Check	Description
	The design identifies, understands, and accommodates the company security policy.
☐	Restrictions imposed by infrastructure security (including available services, protocols, and firewall restrictions) are identified.
☐	The design recognizes and accommodates restrictions imposed by hosting environments (including application isolation requirements).
☐	The target environment code-access-security trust level is known.
☐	The design identifies the deployment infrastructure requirements and the deployment configuration of the application.
☐	Domain structures, remote application servers, and database servers are identified.
☐	The design identifies clustering requirements.
☐	The design identifies the application configuration maintenance points (such as what needs to be configured and what tools are available for an IDC admin) .
☐	Secure communication features provided by the platform and the application are known.
☐	The design addresses Web farm considerations (including session state management, machine specific encryption keys, Secure Sockets Layer (SSL), certificate deployment issues, and roaming profiles).
☐	The design identifies the certificate authority (CA) to be used by the site to support SSL.
☐	The design addresses the required scalability and performance criteria.

Application Architecture and Design Considerations

Input Validation

Check	Description
☐	All entry points and trust boundaries are identified by the design.
☐	Input validation is applied whenever input is received from outside the current trust boundary.
☐	The design assumes that user input is malicious.
☐	Centralized input validation is used where appropriate.
☐	The input validation strategy that the application adopted is modular and consistent.

☐	The validation approach is to constrain, reject, and then sanitize input. (Looking for known, valid, and safe input is much easier than looking for known malicious or dangerous input.)
☐	Data is validated for type, length, format, and range.
☐	The design addresses potential canonicalization issues.
☐	Input file names and file paths are avoided where possible.
☐	The design addresses potential SQL injection issues.
☐	The design addresses potential cross-site scripting issues.
☐	The design does not rely on client-side validation.
☐	The design applies defense in depth to the input validation strategy by providing input validation across tiers.
☐	Output that contains input is encoded using HtmlEncode and UrltEncode.

Authentication

Check	Description
☐	Application trust boundaries are identified by the design.
☐	The design identifies the identities that are used to access resources across the trust boundaries.
☐	The design partitions the Web site into public and restricted areas using separate folders.
☐	The design identifies service account requirements.
☐	The design identifies secure storage of credentials that are accepted from users.
☐	The design identifies the mechanisms to protect the credentials over the wire (SSL, IPSec, encryption and so on)
☐	Account management policies are taken into consideration by the design
☐	The design ensure that minimum error information is returned in the event of authentication failure
☐	The identity that is used to authenticate with the database is identified by the design.
☐	If SQL authentication is used, credentials are adequately secured over the wire (SSL or IPSec) and in storage (DPAPI).
☐	The design adopts a policy of using least-privileged accounts.
☐	Password digests (with salt) are stored in the user store for verification.
☐	Strong passwords are used.
☐	Authentication tickets (cookies) are not transmitted over non-encrypted connections.

Authorization

Check	Description
☐	The role design offers sufficient separation of privileges (the design considers authorization granularity).
☐	Multiple gatekeepers are used for defense in depth.
☐	The application's login is restricted in the database to access-specific stored procedures.
☐	The application's login does not have permissions to access tables directly.
☐	Access to system level resources is restricted.
☐	The design identifies code access security requirements. Privileged resources and privileged operations are identified.
☐	All identities that are used by the application are identified and the resources accessed by each identity are known.

Configuration Management

Check	Description
☐	Administration interfaces are secured (strong authentication and authorization is used).
☐	Remote administration channels are secured.
☐	Configuration stores are secured.
☐	Configuration secrets are not held in plain text in configuration files.
☐	Administrator privileges are separated based on roles (for example, site content developer or system administrator).

Check	
☐	Least-privileged process accounts and service accounts are used.

Sensitive Data

Check	Description
☐	Secrets are not stored unless necessary. (Alternate methods have been explored at design time.)
☐	Secrets are not stored in code.
☐	Database connections, passwords, keys, or other secrets are not stored in plain text.
☐	The design identifies the methodology to store secrets securely. (Appropriate algorithms and key sizes are used for encryption. It is preferable that DPAPI is used to store configuration data to avoid key management.)
☐	Sensitive data is not logged in clear text by the application.
☐	The design identifies protection mechanisms for sensitive data that is sent over the network.
☐	Sensitive data is not stored in persistent cookies.
☐	Sensitive data is not transmitted with the GET protocol.

Session Management

Check	Description
☐	SSL is used to protect authentication cookies.
☐	The contents of authentication cookies are encrypted.
☐	Session lifetime is limited.
☐	Session state is protected from unauthorized access.
☐	Session identifiers are not passed in query strings.

Cryptography

Check	Description
☐	Platform-level cryptography is used and it has no custom implementations.
☐	The design identifies the correct cryptographic algorithm (and key size) for the application's data encryption requirements.
☐	The methodology to secure the encryption keys is identified.
☐	The design identifies the key recycle policy for the application.
☐	Encryption keys are secured.
☐	DPAPI is used where possible to avoid key management issues.
☐	Keys are periodically recycled.

Parameter Manipulation

Check	Description
☐	All input parameters are validated (including form fields, query strings, cookies, and HTTP headers).
☐	Cookies with sensitive data are encrypted.
☐	Sensitive data is not passed in query strings or form fields.
☐	HTTP header information is not relied on to make security decisions.
☐	View state is protected using MACs.

Exception Management

Check	Description
☐	The design outlines a standardized approach to structured exception handling across the application.
☐	Application exception handling minimizes the information disclosure in case of an exception.
☐	The design identifies generic error messages that are returned to the client.
☐	Application errors are logged to the error log.
☐	Private data (for example, passwords) is not logged.

Auditing and Logging

Check	Description
☐	The design identifies the level of auditing and logging necessary for the application and identifies the

key parameters to be logged and audited.

- ☐ The design considers how to flow caller identity across multiple tiers (at the operating system or application level) for auditing.
- ☐ The design identifies the storage, security, and analysis of the application log files.

SQL Injection Checks

Check	Description
☐	Input passed to data access methods that originates outside the current trust boundary is constrained. Sanitization of input is only used as a defense in depth measure.
☐	Stored procedures that accept parameters are used by data access code. If stored procedures are not used, type safe SQL parameters are used to construct SQL commands.
☐	Least-privileged accounts are used to connect to the database.

Authentication

Check	Description
☐	Microsoft® Windows® operating system authentication is used to connect to the database.
☐	Strong passwords are used and enforced.
☐	If Microsoft SQL Server™ authentication is used, the credentials are secured over the network by using IPSec or SSL, or by installing a database server certificate.
☐	If SQL Server authentication is used, connection strings are encrypted by using DPAPI and are stored in a secure location.
☐	Application connects using a least-privileged account. The **sa** account or other privileged accounts that are members of the **sysadmin** or **db_owner** roles are not used for application logins.

Authorization

Check	Description
☐	Calling users are restricted using declarative or imperative principal permission checks (normally performed by business logic).
☐	Calling code is restricted using identity permission demands in scenarios where you know and want to limit the calling code.
☐	Application login is restricted in the database and can only execute selected stored procedures. Application's login has no direct table access.

Configuration Management

Check	Description
☐	Windows authentication is used to avoid credential management.
☐	Connection strings are encrypted and encrypted data is stored securely, for example, in a restricted registry key.
☐	OLE DB connection strings do not contain Persist Security Info="true" or "yes".
☐	UDL files are secured with restricted ACLs.

Sensitive Data

Check	Description
☐	Sensitive data is encrypted in the database using strong symmetric encryption (for example, 3DES).
☐	Symmetric encryption keys are backed up and encrypted with DPAPI and stored in a restricted registry key.
☐	Sensitive data is secured over the network by using SSL or IPSec.
☐	Passwords are not stored in custom user store databases. Password hashes are stored with salt values instead.

Deployment Considerations

Check	Description
☐	Firewall restrictions ensure that only the SQL Server listening port is available on the database server.
☐	A method for maintaining encrypted database connection strings is defined.
☐	The application is configured to use a least-privileged database login.
☐	SQL server auditing is configured. Failed login attempts are logged at minimum.
☐	Data privacy and integrity over the network is provided with IPSec or SSL.

RACI Chart

RACI stands for:

- **R**esponsible (the role responsible for performing the task)
- **A**ccountable (the role with overall responsibility for the task)
- **C**onsulted (people who provide input to help perform the task)
- Keep **I**nformed (people with a vested interest who should be kept informed)

You can use a RACI chart at the beginning of your project to identify the key security related tasks together with the roles that should execute each task.

Table 4 illustrates a simple RACI chart for this guidance. (The heading row lists the roles; the first column lists tasks, and the remaining columns delineate levels of accountability for each task according to role.)

Table 4. RACI Chart

Tasks	Architect	System Administrator	Developer	Tester	Security Professional
Security Policies		R		I	A
Threat Modeling	A		I	I	R
Security Design Principles	A	I	I		C
Security Architecture	A	C			R
Architecture and Design Review	R				A
Code Development			A		R
Technology Specific Threats			A		R
Code Review			R	I	A
Security Testing	C		I	A	C
Network Security	C	R			A
Host Security	C	A	I		R
Application Security	C	I	A		R
Deployment Review	C	R	I	I	A

B. COBIT *APPLICATION DEVELOPMENT CONTROL*

COBIT memiliki standard untuk mengkontrol pada pengembangan suatu system informasi (aplikasi). Adapun isinya terdiri atas:

PO 4 – Define the Information Technology Organization & Relationships

- PO 4.4, Roles & Responsibilities
- PO 4.7, Ownership & Custodianship
- PO 4.10, Segregation of Duty (SOD)

PO 10 - Manage Projects

- PO 10.01, Project Management Framework
- PO 10.02, User Participation in Project Initiation
- PO 10.03, Project Team Membership & Responsibilities
- PO 10.04, Project Definition
- PO 10.05, Project Approval
- PO 10.06, Project Phase Approval
- PO 10.07, Project Master Plan
- PO 10.08, System Quality Assurance Plan
- PO 10.09, Planning of Assurance Methods
- PO 10.10, Formal Project Risk Management
- PO 10.11, Test Plan
- PO 10.12, Training Plan
- PO 10.13, Post-Implementation Review Plan

PO 11 - Manage Quality

- PO 11.05, Systems Development Life Cycle Methodology

PO 11.08, Coordination and Communication

AI 2 - Acquire & Maintain Application Software

- AI 2.01, Design Methods
- AI 2.02, Major Changes to Existing Systems
- AI 2.03, Design Approval
- AI 2.04, File Requirements Definition & Documentation
- AI 2.05, Program Specifications
- AI 2.06, Source Data Collection Design
- AI 2.07, Input Requirements Definition & Documentation
- AI 2.08, Definition of Interfaces
- AI 2.09, User-Machine Interface
- AI 2.10, Processing Requirements Definition & Documentation
- AI 2.11, Output Requirements Definition & Documentation
- AI 2.12, Controllability
- AI 2.13, Availability as a Key Design Factor
- AI 2.14, Integrity Provisions in Application Program Software
- AI 2.15, Application Software Testing

- AI 2.16, User Reference and Support Materials
- AI 2.17, Reassessment of System Design

AI 6 - Manage Change

- AI 6.01, Change Request Initiation & Control
- AI 6.02, Impact Assessment
- AI 6.03, Control of Changes
- AI 6.04, Emergency Changes
- AI 6.05, Documentation & Procedures
- AI 6.06, Authorized Maintenance
- AI 6.07, Software Release Policy

AI 6.08, Distribution of Software

C. ISO 17799

What Is The ISO 17799 Standard?

1. ISO – International Organization for Standardization
2. Complete Set Of Controls To Ensure The Best Practices For Information Security
3. The Major Standard - Internationally Recognized Information Security Standard
4. Guideline - Guiding principle providing a good starting point for implementing information security. They are either based on essential legislative requirements or considered to be common best practices for information security.

- Legislative Controls
  1. 12.1.4 – Data Protection and Privacy of Personal Information
  2. 12.1.3 – Safeguarding of Organizational Records
  3. 12.1.2 – Intellectual Property Rights
- Best Practices
  4. 3.1 – Information Security Policy Document
  5. 4.1.3 – Allocation of Information Security Responsibilities
  6. 6.2.1 – Information Security Education and Training
  7. 6.3.1 – Reporting Security Incidents

8 11.1 Business Continuity Management

10 Sections

- Security Policy – To provide management direction & support for information security
- Organizational Security – Manage information security within the organization
- Asset Classification and Control – To maintain appropriate protection of organizational assets
- Personnel Security – To reduce the risk of human error, theft, fraud or misuse of facilities
- Physical & Environmental Security – To prevent unauthorized access, damage and interference to business premises and information

- Communications and Operations Management – To ensure the correct and secure operations of information processing facilities
- Access Control – Control access to information
- System Development and Maintenance – To ensure security is built into information systems
- Business Continuity Management – To counteract interruptions to business activities and to protect critical business processes from the effects of major failures or disasters
- Compliance – To avoid breaches of any criminal and civil law, statutory, regulatory or contractual

D. CMM