PEMERINTAH DAERAH PROVINSI JAWA BARAT
SEKRETARIAT DAERAH
Jalan Diponegoro Nomor 22 Telepon : (022) 4232448 – 4233347 – 4260963
Faksimil : (022) 4203450 Website : www.jabarprov.go.id e-mail : info@jabarprov.go.id
Bandung - 40115

KEPUTUSAN GUBERNUR JAWA BARAT
NOMOR 473.12/Kep.572-Diskominfo/2024
TENTANG
PEDOMAN TEKNIS SIKLUS HIDUP PENGEMBANGAN PERANGKAT LUNAK YANG AMAN

GUBERNUR JAWA BARAT,

Menimbang : a. bahwa pengembangan perangkat lunak atau aplikasi khusus dalam pemanfaatan teknologi informasi dan komunikasi dalam rangka penyelenggaraan sistem pemerintahan berbasis elektronik di lingkungan Pemerintah Provinsi Jawa Barat, harus memenuhi aspek keamanan informasi pada setiap tahapan dari siklus pengembangan perangkat lunak;

b. bahwa berdasarkan pertimbangan sebagaimana dimaksud dalam huruf a, perlu menetapkan Keputusan Gubernur tentang Pedoman Teknis Siklus Hidup Pengembangan Perangkat Lunak yang Aman;

Mengingat : 1. Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik (Lembaran Negara Republik Indonesia Tahun 2008 Nomor 58, Tambahan Lembaran Negara Republik Indonesia Nomor 4843) sebagaimana telah diubah beberapa kali, terakhir dengan Undang-Undang Nomor 1 Tahun 2024 tentang Perubahan Kedua atas Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik (Lembaran Negara Republik Indonesia Tahun 2024 Nomor 1, Tambahan Lembaran Negara Republik Indonesia Nomor 6905);

2. Undang-Undang Nomor 23 Tahun 2014 tentang Pemerintahan Daerah (Lembaran Negara Republik Indonesia Tahun 2014 Nomor 244, Tambahan Lembaran Negara Republik Indonesia Nomor 5587) sebagaimana telah diubah beberapa kali, terakhir dengan Undang-Undang Nomor 6 Tahun 2023 tentang Penetapan Peraturan Pemerintah Pengganti Undang-Undang Nomor 2 Tahun 2022 tentang Cipta Kerja Menjadi Undang-Undang (Lembaran Negara Republik Indonesia Tahun 2023 Nomor 41, Tambahan Lembaran Negara Republik Indonesia Nomor 6856);

3. Undang-Undang Nomor 10 Tahun 2023 tentang Provinsi Jawa Barat (Lembaran Negara Republik Indonesia Tahun 2023 Nomor 57, Tambahan Lembaran Negara Republik Indonesia Nomor 6866);

4. Peraturan Pemerintah Nomor 71 Tahun 2019 tentang Penyelenggaraan Sistem dan Transaksi Elektronik (Lembaran Negara Republik Indonesia Tahun 2019 Nomor 185, Tambahan Lembaran Negara Republik Indonesia Nomor 6400);

5. Peraturan Presiden Nomor 95 Tahun 2018 tentang Sistem Pemerintahan Berbasis Elektronik (Lembaran Negara Republik Indonesia Tahun 2018 Nomor 182, Tambahan Lembaran Negara Republik Indonesia Nomor 6866);
6. Peraturan Menteri Pendayagunaan Aparatur Negara dan Reformasi Birokrasi Nomor 59 Tahun 2020 tentang Pemantauan dan Evaluasi Sistem Pemerintahan Berbasis Elektronik (Berita Negara Republik Indonesia Tahun 2020 Nomor 994);
7. Peraturan Badan Siber dan Sandi Negara Nomor 4 Tahun 2021 tentang Pedoman Manajemen Keamanan Informasi Sistem Pemerintahan Berbasis Elektronik dan Standar Teknis dan Prosedur Keamanan Sistem Pemerintahan Berbasis Elektronik (Berita Negara Republik Indonesia Tahun 2021 Nomor 541);
8. Peraturan Gubernur Jawa Barat Nomor 161 Tahun 2022 tentang Penyelenggaraan Sistem Pemerintah Berbasis Elektronik di Lingkungan Pemerintah Daerah Provinsi Jawa Barat (Berita Daerah Provinsi Jawa Barat Tahun 2022 Nomor 162);

MEMUTUSKAN:

Menetapkan : KEPUTUSAN GUBERNUR TENTANG PEDOMAN TEKNIS SIKLUS HIDUP PENGEMBANGAN PERANGKAT LUNAK YANG AMAN.

KESATU : Pedoman Teknis Siklus Hidup Pengembangan Perangkat Lunak yang Aman yang selanjutnya disebut Pedoman Teknis, sebagaimana tercantum dalam Lampiran yang merupakan bagian yang tidak terpisahkan dari Keputusan Gubernur ini.

KEDUA : Pedoman Teknis sebagaimana dimaksud dalam Diktum KESATU, menjadi pedoman setiap Perangkat Daerah Provinsi Jawa Barat dalam pengembangan perangkat lunak atau aplikasi khusus dalam rangka penyelenggaran sistem pemerintahan berbasis elektronik.

KETIGA : Keputusan Gubernur ini mulai berlaku pada tanggal ditetapkan.

Ditetapkan di Bandung
pada tanggal 8 Oktober 2024

a.n. GUBERNUR JAWA BARAT
SEKRETARIS DAERAH,

Ditandatangani secara elektronik oleh:
SEKRETARIS DAERAH PROVINSI JAWA BARAT,

Drs. HERMAN SURYATMAN, M.Si.
Pembina Utama

LAMPIRAN
KEPUTUSAN GUBERNUR JAWA BARAT
NOMOR 473.12/Kep.572-Diskominfo/2024
TENTANG
PEDOMAN TEKNIS SIKLUS HIDUP PENGEMBANGAN PERANGKAT LUNAK YANG AMAN.

BAB I

PENDAHULUAN

1.1. Umum

a. Sesuai dengan Peraturan Presiden Nomor 95 Tahun 2018 tentang Sistem Pemerintahan Berbasis Elektronik, Pasal 40 bahwa Pengembangan aplikasi SPBE harus menjamin aspek-aspek keamanan informasi yang meliputi kerahasiaan, keutuhan, ketersediaan, keaslian, dan kenirsangkalan (*non-repudiation*) sumber daya.

b. Untuk menjamin pemenuhan aspek-aspek keamanan informasi, maka diperlukan pedoman teknis untuk keamanan dalam pengembangan aplikasi berupa Pedoman Teknis *Secure Software Development Life Cycle* (SDLC).

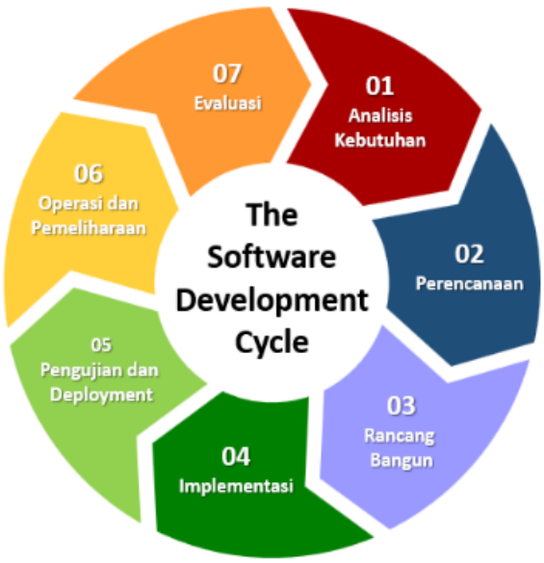
c. Pedoman Teknis *Secure* SDLC memberikan arahan bagi Perangkat Daerah di lingkungan Pemerintah Daerah Provinsi Jawa Barat mengenai hal-hal yang berhubungan dengan aspek keamanan informasi dalam setiap tahapan siklus pengembangan perangkat lunak, baik berbasis *web* maupun *mobile*.

d. *Secure* SDLC merupakan praktik untuk mengintegrasikan aktivitas keamanan (seperti mendefinisikan kebutuhan keamanan, reviu kode program, uji keamanan, dan penilaian risiko) terhadap setiap tahapan dalam proses pengembangan perangkat lunak yang berjalan. Praktik-praktik terbaik di bidang keamanan diimplementasikan bersamaan dengan aspek fungsional pengembangan, termasuk pengamanan lingkungan pengembangan.

e. Pedoman ini mengadopsi tahapan-tahapan sesuai dengan Rancangan Peraturan Menteri Komunikasi dan Informatika tentang Standar Teknis dan Prosedur Pembangunan dan Pengembangan Aplikasi Sistem Pemerintahan Berbasis Elektronik, serta Keputusan Gubernur Jawa Barat Nomor 048/Kep.437-Diskominfo/2023 tentang Standar Teknis dan Prosedur Pembangunan dan Pengembangan Aplikasi Khusus di

Lingkungan Pemerintah Daerah Provinsi Jawa Barat. Tahapan-tahapan tersebut yaitu:

1. Analisis kebutuhan;
2. Perencanaan;
3. Rancang bangun;
4. Implementasi;
5. Pengujian dan deployment;
6. Operasi dan pemeliharaan;
7. Evaluasi.



Gambar 1.1. Siklus Pengembangan Perangkat Lunak

1.2. Maksud dan Tujuan

a. Maksud disusunnya Pedoman Teknis *Secure Software Development Lifecycle* (SDLC) ini adalah untuk memberikan panduan bagi Perangkat Daerah dalam mengintegrasikan praktik-praktik keamanan ke dalam siklus pengembangan perangkat lunak.

b. Adapun tujuannya adalah:

1. Mendeskripsikan tugas-tugas atau teknik pengamanan apa yang harus dilakukan untuk setiap tahapan dalam *Secure* SDLC guna menjamin pengembangan perangkat lunak yang aman.
2. Memberikan penilaian berupa *checklist* untuk tugas-tugas atau teknik tersebut sebagai pedoman ringkas dalam mengimplementasikan Secure SDLC.
3. Memberikan gambaran kondisi keamanan dari perangkat lunak yang akan, sedang, dan telah dibangun.

4. Membantu stakeholders dalam mengetahui dan mengkomunikasikan kondisi keamanan perangkat lunak beserta risikonya.

1.3. Ruang Lingkup

Ruang Lingkup Pedoman Teknis Secure SDLC meliputi:

1. Edukasi, Manajemen, dan Kesadaran;
2. Analisis Kebutuhan;
3. Inisiasi/Perencanaan Proyek;
4. Rancang Bangun;
5. Implementasi;
6. Pengujian dan Deployment;
7. Operasi dan Pemeliharaan; dan
8. Evaluasi.

1.4. Manfaat Penerapan Secure SDLC

Manfaat penerapan Secure SDLC adalah:

1. Meningkatkan keamanan dan kualitas sistem secara keseluruhan.
2. Membuat biaya pengembangan perangkat lunak menjadi lebih efisien.
3. Menjamin keamanan pengguna.
4. Meningkatkan kepercayaan dan kepuasan pengguna serta *stakeholders*, yang berdampak terhadap semakin meningkatnya reputasi layanan.
5. Menciptakan budaya sadar keamanan bagi pengembang dan pengguna, sehingga risiko-risiko dalam layanan dapat dikurangi.
6. Mengoptimalkan pengembangan perangkat lunak serta pengambilan keputusan berkaitan dengan keamanan sudah direncanakan dari awal.
7. Meminimalkan *patches* atau *update* terkait dengan keamanan dalam siklus pengembangan perangkat lunak.

1.5. Penerapan Pedoman Teknis Secure SDLC

a. Setiap tahapan dalam siklus pengembangan perangkat lunak diatur dalam daftar periksa atau *checklist* yang disusun untuk memastikan setiap langkah pengembangan mengikuti prinsip keamanan yang ditetapkan. Ketika Perangkat Daerah mengembangkan perangkat lunak dan mencapai tahapan tertentu, *checklist* yang relevan dapat digunakan sebagai panduan untuk memastikan kesesuaian dengan praktik keamanan yang ditetapkan.

b. Pengadopsian daftar periksa tergantung pada pendekatan pengelolaan risiko yang diambil oleh pengembang. Pengembang perlu mengevaluasi relevansi, kelayakan, dan efektivitas dari setiap praktik keamanan yang ada dalam mitigasi ancaman selama siklus pengembangan perangkat lunak. Misalnya, untuk aplikasi yang hanya berfungsi sebagai situs web informatif tanpa interaksi atau transaksi, daftar periksa terkait autentikasi pengguna tidak berlaku. Contoh lainnya adalah *checklist* yang berkaitan dengan penggunaan komponen *third-party*, hanya relevan jika aplikasi tersebut menggunakan komponen dari pihak ketiga.

c. Beberapa *checklist* diambil dari ketentuan yang diatur dalam Peraturan Badan Siber dan Sandi Negara Nomor 4 Tahun 2021 dan ditandai sebagai berikut:

   ☑ warna hijau untuk daftar periksa terkait keamanan data dan informasi;

   ☑ warna jingga untuk daftar periksa terkait keamanan aplikasi SPBE berbasis web;

   ☑ warna biru untuk daftar periksa terkait keamanan aplikasi SPBE berbasis mobile.

d. Beberapa *checklist* diberi keterangan tambahan "PABW" dan "PABM". Keterangan ini menunjukkan bahwa pedoman atau informasi lebih lanjut mengenai checklist tersebut dapat dilihat dalam dokumen Pedoman Teknis Pengamanan Aplikasi Berbasis Website (PABW) dan Pedoman Teknis Pengamanan Aplikasi Berbasis Mobile (PABM).

1.6. Sistematika

Dokumen ini disusun dengan Sistematika sebagai berikut:

1. BAB I PENDAHULUAN
2. BAB II EDUKASI, MANAJEMEN, DAN KESADARAN
3. BAB III ANALISIS KEBUTUHAN
4. BAB IV INISIASI/PERENCANAAN PROYEK
5. BAB V RANCANG BANGUN
6. BAB VI IMPLEMENTASI
7. BAB VII PENGUJIAN DAN DEPLOYMENT
8. BAB VIII OPERASI DAN PEMELIHARAAN
9. BAB IX EVALUASI
10. BAB X PENUTUP

BAB II
EDUKASI, MANAJEMEN, DAN KESADARAN

2.1. Penyelenggaraan Pelatihan/Edukasi

Pelatihan atau edukasi mengenai keamanan siber bagi tim pengembang diselenggarakan dengan memperhatikan hal-hal sebagai berikut:

1. Sebagian anggota tim pengembang mungkin memiliki latar belakang pendidikan non-IT.
2. Meskipun anggota tim pengembang berlatar belakang IT, pendidikan formal yang telah ditempuhnya mungkin belum mencakup materi mengenai keamanan siber.
3. Jenis serangan, alat bantu keamanan, bahasa pemrograman, dan pengalaman dalam bidang siber terus berkembang, sehingga diperlukan penyegaran pengetahuan bagi anggota tim pengembang untuk mengikuti perkembangan.

2.2. Manajemen Pengembangan Perangkat Lunak yang Aman

Proses pengembangan perangkat lunak yang aman harus memperhatikan berbagai aspek yang dapat mempengaruhi kualitas proses dan hasil yang diperoleh, diantaranya adalah prinsip-prinsip umum dalam merancang keamanan informasi dan penerapan keamanan dalam proses pengembangan model *agile*.

2.2.1 Prinsip-prinsip Umum dalam Perancangan Keamanan Informasi

Hal-hal yang disarankan dalam *Secure* SDLC mengacu kepada prinsip-prinsip umum dalam perancangan keamanan informasi. Keamanan informasi merupakan pemeliharaan/ perlindungan terhadap kerahasiaan (*confidentiality*), integritas, dan ketersediaan (*availability*) informasi. Berikut adalah prinsip-prinsip umum dalam merancang keamanan informasi menurut Saltzer and Schroeder serta *National Centers of Academic Excellence in Information Assurance/Cyber Defense* (NCAE):

1. Ekonomi mekanisme (*economy of mechanism*)

   Desain atau perancangan perangkat lunak harus dijaga agar tetap sederhana dan minimalis, sehingga memudahkan proses pengujian dan verifikasi secara menyeluruh. Desain yang sederhana cenderung memiliki lebih sedikit potensi celah keamanan, serta lebih mudah untuk diperbarui atau diganti sesuai kebutuhan operasional dan keamanan di masa mendatang.

2. Default yang aman (*fail-safe defaults*)

   Hak akses yang diberikan harus didasarkan pada perizinan (*permission*) khusus. Secara default, akses harus dibatasi atau diminimalisir, dan skema perlindungan harus menetapkan kondisi tertentu yang secara eksplisit mengizinkan akses sesuai dengan kebijakan keamanan yang berlaku.

3. Mediasi lengkap (*complete mediation*)

   Setiap akses terhadap fitur atau objek tertentu harus melalui proses otorisasi yang diperiksa secara ketat guna memastikan bahwa hanya pihak yang berwenang yang dapat mengaksesnya.

4. Desain terbuka (*open design*)

   Desain sistem tidak bersifat rahasia. Mekanisme keamanan tidak bergantung pada ketidaktahuan calon penyerang tentang desain sistem, melainkan pada pengamanan melalui kepemilikan kunci atau password yang bersifat rahasia.

5. Pemisahan hak akses (*separation of privilege*)

   Pemisahan hak akses adalah praktik membagi hak akses di antara beberapa entitas untuk mengurangi risiko yang disebabkan oleh peretasan atau penyalahgunaan.

6. Hak minimal (*least privilege*)

   Setiap pengguna atau komponen sistem hanya diberikan hak akses secukupnya untuk menyelesaikan tugasnya.

7. Mekanisme umum seminimal mungkin (*least common mechanism*)

   Praktik ini meminimalkan penggunaan mekanisme umum untuk mengakses sumber daya.

8. Penerimaan psikologis (*psychological acceptability*)

   Antarmuka pengguna harus dirancang agar mudah digunakan dan tidak mengganggu, sehingga pengguna secara otomatis dan rutin dapat menerapkan mekanisme keamanan dengan benar. Mekanisme keamanan harus transparan dan tetap memenuhi kebutuhan keamanan tanpa memberatkan pengguna.

9. Isolasi (*Isolation*)

   Isolasi adalah pemisahan komponen atau entitas dalam sistem untuk melindungi komponen lain jika terjadi peretasan atau kegagalan.

10. Enkapsulasi (*Encapsulation*)

    Enkapsulasi mengacu pada isolasi khusus berdasarkan fungsionalitas *object-oriented*. Prosedur dan data di-enkapsulasi sehingga hanya prosedur di dalam subsistem yang dapat mengakses struktur internalnya.

11. Modularitas

    Modularitas dalam keamanan berarti memisahkan fungsi keamanan sebagai modul terpisah, atau menggunakan arsitektur modular dalam desain dan implementasi. Hal ini memungkinkan *upgrade* bagian desain keamanan tanpa mengubah keseluruhan sistem, dan menyediakan fungsi keamanan umum seperti kriptografi.

12. *Layering* (*defense in depth*)

    Pendekatan keamanan ini menggunakan perlindungan berlapis yang mencakup manusia, teknologi, dan aspek operasional dari sistem informasi, untuk mengurangi risiko yang ada.

13. *Least astonishment*

    Program atau antarmuka pengguna harus beroperasi dengan cara yang dapat diprediksi oleh pengguna, menghindari reaksi yang mengejutkan atau membingungkan.

14. *Zero trust*

    Pendekatan keamanan ini mengasumsikan bahwa tidak ada entitas, baik di dalam maupun di luar jaringan atau sistem organisasi, yang dapat dipercaya sepenuhnya. Kepercayaan harus selalu diverifikasi secara ketat di setiap akses.

2.2.2 Penerapan Keamanan dalam Model *Agile*

Pada pengembangan perangkat lunak dengan model proses *agile*, satu siklus pengembangan sangat singkat (2-6 minggu) sehingga memasukkan unsur keamanan dan melakukan pengujian keamanan menjadi isu tersendiri. Berikut ini adalah beberapa saran untuk penerapan keamanan dalam model *agile*:

1. *Every-sprint approach* dalam Scrum. Pendekatan ini memasukkan unsur keamanan ke dalam *user stories* di setiap *sprint*. Namun, keberhasilan metode ini memerlukan anggota tim pengembangan yang memiliki pengetahuan mendalam tentang keamanan informasi.
2. *Security sprint approach* dalam Scrum. Pada pendekatan ini, *user stories* dan elemen keamanan dianalisis dan dibangun dalam *sprint* yang terpisah. Risiko utama dari metode ini adalah potensi penundaan dalam proses pengembangan perangkat lunak.

3. *Secure scrum*. Variasi dari framework Scrum ini berfokus pada pengembangan perangkat lunak yang aman sepanjang seluruh siklus pengembangan. Inti dari pendekatan ini adalah penggunaan *S-Mark* (penanda pada *user stories* yang mengandung isu keamanan) dan *S-Tag* (penanda yang menunjukkan aspek keamanan tertentu).
4. *Role-based Extreme Programming*. Pendekatan ini menambahkan peran baru dalam tim, yaitu "Security Master." Anggota tim dengan peran ini adalah programmer yang bertanggung jawab untuk fokus pada elemen keamanan dan memastikan keamanan perangkat lunak selama proses pengembangan.
5. *Inbuilt security* dalam *Extreme Programming*. Elemen keamanan diintegrasikan ke dalam sebelas praktik yang ada dalam *Extreme Programming Framework*, memastikan bahwa keamanan menjadi bagian integral dari setiap tahapan pengembangan.

2.3. Peningkatan Kesadaran Keamanan Informasi

Kesadaran keamanan informasi mengacu pada pengetahuan dan sikap individu dalam melindungi aset informasi dan fisik organisasi. Peningkatan kesadaran keamanan menjadi sangat penting karena organisasi perlu dilindungi dari serangan yang dapat menyebabkan kebocoran data. Bagi pengembang perangkat lunak, memahami risiko umum keamanan aplikasi adalah langkah awal untuk mengubah budaya organisasi dan tim, sehingga dapat menghasilkan perangkat lunak yang lebih aman. Berikut adalah 10 risiko keamanan aplikasi menurut OWASP (*Open Web Application Security Project*):

1. *Broken Access Control*

   Kendali akses yang gagal menyebabkan pengguna dapat bertindak di luar wewenang yang diberikan. Hal ini bisa mengakibatkan pengungkapan, modifikasi, atau kerusakan data. Contohnya:

   - Melanggar prinsip *least privilege* atau *deny by default*, yang memungkinkan akses oleh pengguna yang tidak berwenang.
   - *Bypassing* kendali akses melalui modifikasi URL, halaman HTML, atau panggilan API.
   - Mengizinkan akses terhadap akun seseorang melalui parameter id (referensi objek secara langsung).
2. Kegagalan Kriptografi

   Data sensitif seperti kata sandi, nomor kartu kredit, dan informasi pribadi harus dienkripsi. Contoh kesalahan:

   - Pengiriman data dalam bentuk *clear text*.

- Penggunaan algoritma atau protokol kriptografi yang sudah kadaluwarsa.
- Tidak melakukan validasi sertifikat keamanan.

3. Injeksi (Termasuk Cross-Site Scripting / XSS)

   Aplikasi rentan terhadap serangan injeksi jika data dari pengguna tidak divalidasi. Contoh kerentanannya:

   - Data yang diterima dari pengguna tidak divalidasi, difilter, atau dibersihkan (sanitized) terlebih dahulu.
   - Penggunaan perintah dinamis tanpa batasan konteks.
   - Data berbahaya yang digunakan dalam ORM atau SQL yang dapat dieksploitasi.

4. Desain yang Tidak Aman

   Desain yang tidak aman merupakan pengertian luas untuk merepresentasikan berbagai kelemahan yang diakibatkan oleh tidak adanya atau tidak efektifnya desain/kontrol keamanan. Sementara itu, desain yang aman merupakan budaya dan metodologi yang secara rutin mengevaluasi dan memastikan bahwa kode telah dirancang dan diuji untuk mencegah berbagai teknik serangan. Desain yang aman masih memiliki potensi cacat implementasi yang menyebabkan kerentanan dan dapat dieksploitasi.

5. Kesalahan Konfigurasi Keamanan

   Kerentanan aplikasi akibat konfigurasi yang tidak tepat dapat terjadi akibat berbagai faktor, contohnya:

   - Tidak melakukan *security hardening* pada aplikasi.
   - Mengaktifkan fitur yang tidak diperlukan.
   - Menggunakan akun dan kata sandi default yang tidak diubah.

6. Komponen yang Rentan dan Kadaluwarsa

   Risiko muncul jika komponen perangkat lunak tidak diperbarui. Contohnya:

   - Tidak mengetahui versi komponen yang digunakan.
   - Penggunaan perangkat lunak yang tidak lagi didukung.
   - Tidak melakukan *scan* kerentanan secara teratur dan tidak berlangganan buletin/berita keamanan yang berkaitan dengan komponen yang digunakan.

7. Kegagalan Identifikasi dan Autentikasi

   Sistem autentikasi yang lemah membuat sistem rentan terhadap serangan, seperti:

   - *Credential stuffing*, di mana penyerang menggunakan kredensial yang dicuri.
   - Serangan *brute force* yang diizinkan akibat autentikasi yang lemah.
   - Mengizinkan *password default*, lemah, atau yang umum diketahui, seperti “admin/admin”.

8. Kegagalan perangkat lunak dan data dalam hal integritas

   Aplikasi yang bergantung pada plugin atau modul yang tidak diverifikasi dapat merusak integritas data. Fitur *auto-update* yang tidak diverifikasi juga berisiko.

9. Kegagalan Pemantauan dan *Log* Keamanan

   Tanpa *logging* dan *monitoring*, serangan tidak dapat dideteksi. Contohnya adalah:

   - Kegagalan dalam mencatat peristiwa penting seperti login dan transaksi besar.
   - Peringatan dan kesalahan tidak atau kurang memberikan pesan *log* yang cukup jelas.
   - Penyimpanan log secara lokal tanpa pemantauan yang tepat.

10. *Server-side Request Forgery* (SSRF)

    SSRF terjadi ketika aplikasi web mengakses sumber daya remote tanpa memvalidasi URL, memungkinkan penyerang memanipulasi permintaan yang dikirimkan oleh aplikasi.

Daftar periksa edukasi, manajemen, dan kesadaran keamanan informasi mencakup langkah-langkah penting untuk memastikan bahwa seluruh tim pengembangan perangkat lunak memiliki pemahaman yang mendalam mengenai keamanan informasi.

Tabel 2.1. Daftar Periksa Tahap Edukasi, Manajemen, dan Kesadaran

No.	Item	Pertanyaan	Applied	Done
A. Penyelenggaraan Pelatihan				
1	Pelaksanaan pelatihan	Apakah anggota tim mendapatkan pelatihan atau penyegaran dalam pengetahuan mengenai keamanan siber secara teratur minimal 1 tahun sekali?	☐	☐
2	Relevansi pelatihan	Apakah pelatihan direncanakan dan dilaksanakan sesuai dengan kebutuhan serta peran anggota tim?	☐	☐
3	Rencana *outcome*	Apakah *outcome* yang diinginkan dari pelatihan direncanakan sebelumnya?	☐	☐
4	Evaluasi *outcome*	Apakah outcome yang diinginkan dari pelatihan dievaluasi sesudahnya?	☐	☐
5	Ketercukupan pelatihan	Apakah pegawai dilatih dengan cukup untuk untuk melaksanakan tugas dan tanggung jawab dalam keamanan informasi?	☐	☐
B. Manajemen Pengembangan Perangkat Lunak yang Aman				
B.1. Tim dan Personil				
1	Kepercayaan terhadap individu	Apakah individu yang menduduki posisi untuk tanggung jawab tertentu dalam organisasi (termasuk pihak ketiga penyedia layanan) dapat dipercaya dan memenuhi kriteria keamanan yang ditetapkan?	☐	☐

No.	Item	Pertanyaan	Applied	Done
2	Perubahan pegawai	Apakah sistem dan informasi dilindungi selama dan sesudah perubahan dalam kepegawaian seperti pemberhentian dan penerimaan pegawai baru?	☐	☐
3	Sanksi pelanggaran	Apakah ada sanksi resmi bagi personil yang melanggar untuk kepatuhan dalam kebijakan dan prosedur keamanan organisasi?	☐	☐
4	Keamanan pihak ketiga	Apakah sudah dipastikan bahwa penyedia pihak ketiga menerapkan ukuran keamanan yang cukup untuk melindungi informasi, aplikasi, dan/atau layanan yang di-*outsource*-kan?	☐	☐
5	Latar belakang pengembang	Apakah sudah diperiksa latar belakang personil atau pihak ketiga yang membangun sistem, baik dari sisi kompetensi maupun integritas?	☐	☐
6	Evaluasi pengembang	Apakah telah dilakukan pemantauan dan evaluasi terhadap personil atau pihak ketiga yang mengembangkan sistem?	☐	☐
B.2. Proses Secure SDLC				
1	Pendokumen-tasian	Apakah proses pengembangan yang berkaitan dengan keamanan sudah didokumentasikan sepanjang siklus hidup perangkat lunak?	☐	☐

No.	Item	Pertanyaan	Applied	Done
2	Pengelolaan temuan	Apakah berbagai temuan keamanan sudah dikelola dan dapat dilacak?	☐	☐
3	Tindak lanjut temuan	Apakah sudah dilakukan aksi tertentu sebagai tindak lanjut temuan?	☐	☐
4	Perlindungan media	Apakah sudah ada mekanisme perlindungan terhadap media, baik kertas maupun digital?	☐	☐
5	Pembuangan dan penggunaan kembali media	Apakah media sudah dibersihkan atau dihancurkan datanya sebelum dibuang atau dipakai kembali?	☐	☐
6	Manajemen perubahan (*change management*)	Apakah sudah diterapkan manajemen perubahan untuk memastikan bahwa perubahan terhadap aset atau sistem informasi telah diuji, direviu, dan disetujui, termasuk analisis kemungkinan kerentanan baru akibat perubahan tersebut?	☐	☐
7	Fitur kendali versi (*versioning* kode sumber)	Apakah sudah digunakan fitur kendali versi (*version control features*) pada *repository* untuk melacak perubahan, memastikan tim bekerja pada versi yang benar, dan memberikan kemampuan untuk kembali ke versi sebelumnya jika dibutuhkan?	☐	☐
8	Reviu setiap tahapan	Apakah dilaksanakan reviu formal/informal pada setiap akhir	☐	☐

No.	Item	Pertanyaan	Applied	Done
		tahapan pengembangan perangkat lunak?		
9	Manajemen konfigurasi	Apakah konfigurasi yang aman terhadap *hardware* (perangkat *end point* dan perangkat jaringan) dan software (sistem operasi, database, aplikasi) beserta komponennya telah ditetapkan, disimpan, dan direviu?	☐	☐
10	Evaluasi berkala terhadap pengecualian	Apakah pengecualian terhadap kebutuhan atau kebijakan keamanan (misalnya diperbolehkan menggunakan *shared id* untuk sementara waktu) dievaluasi secara berkala?	☐	☐
11	Integrasi proses bisnis	Apakah beberapa praktik dalam Secure SDLC yang relevan juga diintegrasikan dalam proses bisnis organisas (contoh: analisis risiko, security awareness)?	☐	☐
B.3. Teknologi				
1	Pengelolaan *shadow* IT	Apakah dilakukan pengendalian dan pemantauan terhadap *shadow* IT (*hardware* dan *software* yang tidak dikelola oleh bagian IT)?	☐	☐
2	Teknologi *refresh management*	Apakah *hardware* dan *software* yang digunakan masih belum melebihi *end-of-support* (EOS) dari vendor?	☐	☐
3	*Endpoint protection*	Apakah perangkat *client* dan *server* yang digunakan dalam	☐	☐

No.	Item	Pertanyaan	Applied	Done
		pengembangan, operasional, dan pemeliharaan sudah dilindungi dengan anti-malware dan secara teratur dilakukan *scanning*?		
4	Pengelolaan *log*	Apakah log sudah dipantau dan dievaluasi secara berkala, serta dilindungi dari akses tidak sah?	☐	☐
5	Penyimpanan di *repository*	Apakah kode sumber dan *configuration-as-code* sudah disimpan dalam *repository*, serta sudah dibatasi aksesnya?	☐	☐
B.4. Kebijakan				
1	Rencana pengarsipan	Apakah sudah didefinisikan rencana/kebijakan mengenai apa yang harus diarsipkan untuk setiap rilis perangkat lunak (misalnya kode program, dokumentasi, data, *third-party libraries*, *package files*)?	☐	☐
2	Implementasi pengarsipan	Apakah pengarsipan untuk setiap rilis perangkat lunak telah dilakukan sesuai dengan rencana/kebijakan?	☐	☐
3	Saluran komunikasi	Apakah sudah ditentukan saluran komunikasi yang aman untuk anggota tim dalam berdiskusi?	☐	☐
4	Definisi keberhasilan	Apakah ukuran keberhasilan pengembangan perangkat lunak dari sisi keamanan sudah didefinisikan (misalnya: hasil pengelolaan risiko, perbaikan	☐	☐

No.	Item	Pertanyaan	Applied	Done
		kerentanan, atau ukuran lainnya)?		
5	Konfigurasi lingkungan	Apakah konfigurasi lingkungan pengembangan dan operasional berdasarkan konfigurasi *hardware* dan *software* yang telah ditetapkan/direncanakan?	☐	☐
B.5. Penerapan Prinsip Keamanan				
1	*Zero trust architeture*	Apakah konfigurasi lingkungan pengembangan dan operasional menerapkan prinsip *zero trust architecture*?	☐	☐
2	*Defense in depth*	Apakah sistem memiliki pertahanan berlapis?	☐	☐
C. Peningkatan Kesadaran dan Pengetahuan Risiko Umum Keamanan Aplikasi				
1	Menyadari 10 risiko keamanan aplikasi	Apakah tim pengembang telah mengetahui dan menyadari mengenai 10 risiko umum keamanan aplikasi?	☐	☐
2	Kesadaran risiko dari manajer	Apakah manajer dan pengguna sadar akan risiko keamanan yang berhubungan dengan aktivitas mereka serta terhadap hukum, regulasi, dan kebijakan berkaitan dengan keamanan?	☐	☐
3	Edukasi perubahan kebutuhan	Apakah jika ada perubahan pada kebutuhan keamanan dan implementasinya, hal tersebut diinformasikan kepada individu/pengguna yang terimbas?	☐	☐

No.	Item	Pertanyaan	Applied	Done
4	Diseminasi kebijakan keamanan	Apakah pengembang aplikasi, baik internal maupun rekanan telah mendapatkan pelatihan atau diseminasi mengenai kebijakan keamanan informasi dan pedoman teknis Secure SDLC?	☐	☐
D. Pengelolaan Risiko				
1	Identifikasi dan penilaian risiko	Apakah sudah dilakukan identifikasi dan penilaian (*assessment*) secara periodik mengenai risiko operasional organisasi (termasuk misi, fungsi, citra, atau reputasi), aset, dan individu, yang berhubungan dengan sistem informasi serta pemrosesan, penyimpanan, atau transmisi yang berkaitan?	☐	☐
2	Pertimbangan laporan sebelumnya	Jika perangkat lunak merupakan versi baru, apakah penilaian risiko sudah mempertimbangkan laporan kerentanan atau insiden pada versi sebelumnya?	☐	☐
3	Risiko *technology stacks*	Apakah sudah dilakukan analisis risiko terhadap *technology stacks* yang diterapkan (bahasa pemrograman, lingkungan pengembangan, model *deployment*)?	☐	☐
4	Area risiko	Apakah telah dilakukan penilaian / analisis untuk area-area tertentu yang berisiko cukup	☐	☐

No.	Item	Pertanyaan	Applied	Done
		tinggi seperti autentikasi, pengelolaan id pengguna atau *password*, dan *access control*?		
5	Penerimaan risiko	Apakah risiko keamanan yang diterima sudah sesuai dengan nilai dampak dan ruang lingkupnya?	☐	☐
6	Penanganan/ mitigasi risiko	Apakah sudah dilakukan penanganan/*treatment*/mitigasi terhadap risiko yang telah diidentifikasi dan dinilai?	☐	☐
7	Pencatatan risiko dan *treatment*-nya	Apakah identifikasi, penilaian, dan penanganan risiko telah dicatat sehingga dapat digunakan untuk kebutuhan audit dan pemeliharaan?	☐	☐
8	Pemantauan, reviu/evaluasi, dan pelaporan	Apakah sudah dilakukan pemantauan, reviu/evaluasi, dan pelaporan terhadap risiko yang ada?	☐	☐
9	Pengelolaan risiko *data center*	Untuk perangkat daerah yang memiliki data center sendiri, apakah telah dilaksanakan penilaian risiko terhadap ancaman dan kerentanan *data center*?	☐	☐

BAB III
ANALISIS KEBUTUHAN

Umumnya istilah analisis kebutuhan dalam rekayasa perangkat lunak mengacu kepada proses untuk menginventarisasi keinginan atau harapan pengguna akan produk yang akan dibangun. Namun pada dokumen ini, analisis kebutuhan menyesuaikan dengan apa yang dimaksud dalam Rancangan Peraturan Menteri Komunikasi dan Informatika tentang Standar Teknis dan Prosedur Pembangunan dan Pengembangan Aplikasi Sistem Pemerintahan Berbasis Elektronik, serta Keputusan Gubernur Jawa Barat Nomor: 048/Kep.437-Diskominfo/2023 tentang Standar Teknis dan Prosedur Pembangunan dan Pengembangan Aplikasi Khusus di Lingkungan Pemerintah Daerah Provinsi Jawa Barat.

Analisis kebutuhan dalam kedua dokumen tersebut lebih mengacu kepada uraian permasalahan dan kebutuhan yang melatarbelakangi pembangunan/pengembangan aplikasi, termasuk dasar hukum, analisis biaya dan manfaat aplikasi, analisis risiko, dan aspek-aspek lainnya. Untuk pengertian analisis kebutuhan yang umum digunakan dalam konteks rekayasa perangkat lunak, pada dokumen pedoman ini akan disebut sebagai spesifikasi kebutuhan. Tahap pendefinisian spesifikasi kebutuhan dibahas pada Bab 5 Rancang Bangun.

Tidak ada kekhususan dalam hal keamanan pada tahap analisis kebutuhan ini. Meskipun demikian, uraian mengenai analisis risiko dapat menyertakan berbagai kemungkinan yang berkaitan dengan keamanan. Detail mengenai risiko keamanan akan dibahas pada Lampiran II, dimana pengelolaan risiko dilaksanakan sepanjang daur hidup siklus pengembangan perangkat lunak.

BAB IV
INISIASI/PERENCANAAN PROYEK

Pada tahap inisiasi, organisasi mendefinisikan tujuan dan kebutuhan umum dari sistem yang akan dibangun. Tahap ini juga mencakup penentuan apakah perangkat lunak tersebut merupakan sistem informasi yang berdiri sendiri atau merupakan bagian dari sistem yang telah ada. Penilaian risiko awal, termasuk risiko keamanan, biasanya dilakukan pada tahap ini. Selain itu, rencana awal terkait keamanan sistem perlu dipertimbangkan dengan cermat, khususnya dalam menentukan sejauh mana aspek keamanan akan mempengaruhi biaya dan waktu penyelesaian proyek. Cakupan kegiatan tahap inisiasi/perencanaan proyek meliputi:

1. Perencanaan pengembangan;
2. Pemakaian *tools* yang ditetapkan;
3. Pengamanan lingkungan pengembangan;
4. Identifikasi kasus penyalahgunaan (*abuse cases*); dan
5. Penentuan kebutuhan/persyaratan keamanan.

4.1. Perencanaan Pengembangan

Secure Software Development Life Cycle (SDLC) yang efektif tidak hanya mencakup berbagai praktik pengamanan perangkat lunak, tetapi juga mengintegrasikan praktik-praktik tersebut ke dalam proses bisnis dan manajemen organisasi secara menyeluruh. Hal ini mencakup manajemen program, manajemen pemangku kepentingan (*stakeholder management*), perencanaan *deployment*, pengukuran serta indikator kinerja, dan rencana perbaikan berkelanjutan. Perencanaan pengembangan yang aman dalam Secure SDLC merupakan langkah penting untuk memastikan bahwa perangkat lunak yang dibangun memiliki tingkat keamanan yang memadai dan terlindungi dari berbagai ancaman serta risiko keamanan.

4.2. Pemakaian *Tools* yang Ditetapkan

Perangkat Daerah wajib memastikan keamanan tools yang digunakan dalam pengembangan aplikasi. Perangkat Daerah menetapkan tools yang diizinkan untuk digunakan dan harus mematuhi ketentuan pemakaian tools tersebut. Dalam hal penggunaan versi, disarankan untuk selalu menggunakan versi terbaru, karena versi terbaru umumnya dilengkapi dengan fungsi keamanan yang diperbarui dan perlindungan tambahan sesuai dengan perkembangan ancaman keamanan terkini.

4.3. Pengamanan Lingkungan Pengembangan

Pengamanan lingkungan pengembangan dan pengujian bertujuan melindungi kode sumber dari akses pihak yang tidak berwenang serta menjamin integritasnya. Pengamanan ini mencakup pengelolaan dan perlindungan terhadap infrastruktur, perangkat, alat, serta data yang digunakan selama proses pengembangan perangkat lunak, untuk memastikan bahwa kode tetap aman dan tidak dimodifikasi secara tidak sah.

4.4. Identifikasi Kasus Penyalahgunaan (*Abuse Cases*)

Identifikasi kasus penyalahgunaan (*abuse cases*) membantu pengembang untuk memahami perangkat lunak dari perspektif penyerang, dengan tujuan menentukan bagaimana perangkat lunak akan bereaksi terhadap potensi serangan. Pola serangan digunakan untuk memodelkan berbagai skenario penyalahgunaan, seperti serangan di mana aplikasi e-mail gratis disusupi oleh trojan.

Use case menggambarkan perilaku normal dari sistem, sementara *abuse case* berfokus pada aktor jahat yang berupaya menyerang sistem. Analis mendefinisikan tipe-tipe aktor yang berpotensi menyerang dan membuat satu atau lebih *abuse cases* untuk setiap aktor. Analis kemudian mempelajari interaksi antara *use case* dan *abuse case* untuk merancang pertahanan sistem. Sebagai contoh, pada sebuah mobil, *use case* untuk pengemudi adalah 'mengemudi mobil', sementara *abuse case* dari pencuri adalah 'mencuri mobil'. Untuk memitigasi ancaman ini, *use case* baru, seperti 'mengunci mobil', dapat ditambahkan untuk melindungi sistem.

Kesalahan manusia juga berkontribusi terhadap banyak kasus peretasan. Misalnya, analis sistem harus mempertimbangkan pengguna yang menjadi korban *phishing* sebagai *misuse case*, yang perlu dianalisis dengan pendekatan serupa dengan *abuse case*. Hasil analisis dari serangan, baik dari *abuse cases* maupun *misuse cases*, menjadi input untuk tahap lain dalam pengembangan, seperti mendefinisikan kebutuhan keamanan, melakukan *penetration testing*, dan pengujian keamanan berbasis risiko.

4.5. Penentuan Kebutuhan/Persyaratan Keamanan

Kebutuhan keamanan inti mencakup kerahasiaan (*confidentiality*), integritas (*integrity*), ketersediaan (*availability*), autentikasi (*authentication*), otorisasi (*authorization*), serta kenirsangkalan (*non-repudiation*):

1. Kerahasiaan (*confidentiality*)

   Kerahasiaan bertujuan melindungi data pribadi atau sensitif dari pengungkapan yang tidak sah. Mekanisme perlindungan meliputi kriptografi atau *masking*, di mana *masking* adalah metode perlindungan yang lebih lemah (misalnya, mengganti informasi dengan tanda bintang "*"). Perlindungan kerahasiaan harus diterapkan saat pengiriman, pemrosesan, dan penyimpanan data.
2. Integritas (*integrity*)

   Integritas memastikan bahwa data dan program hanya dapat diubah oleh pihak yang berwenang dan melalui cara yang telah disetujui. Mekanisme perlindungan integritas termasuk pencegahan serangan injeksi SQL, validasi input, hashing, pemeriksaan redundansi siklik (CRC), dan manajemen *resource locking* dalam basis data.
3. Ketersediaan (*availability*)

   Ketersediaan memastikan layanan tidak terganggu dan tetap dapat diakses. Perlindungan ini melibatkan *backup*, pencegahan serangan *DoS*, *load balancing*, pengaturan Disaster Recovery Center (DRC), serta pengujian skenario bencana dan pengalihan otomatis (failover) atau manual (switchover) ke sistem redundan.
4. Autentikasi (*authentication*)

   Autentikasi memastikan keabsahan identitas yang diklaim oleh entitas. Bentuk autentikasi meliputi *forms authentication*, *token-based authentication*, dan *smart card-based authentication*. Autentikasi multifaktor, seperti OTP (*One Time Password*), sering digunakan untuk meningkatkan keamanan autentikasi.
5. Otorisasi (*authorization*)

   Otorisasi mengonfirmasi apakah entitas memiliki hak yang diperlukan untuk mengakses sumber daya atau melakukan tindakan tertentu. Model kontrol akses yang digunakan meliputi Role-Based Access Control (RBAC), Discretionary Access Control (DAC), dan Mandatory Access Control (MAC).

6. Kenirsangkalan (*non-repudiation*)

   Kenirsangkalan memastikan bahwa pihak yang melakukan tindakan tertentu, terutama dalam transaksi atau komunikasi elektronik, tidak dapat menyangkal tindakan tersebut. Praktik yang digunakan meliputi tanda tangan elektronik, *timestamping*, log aktivitas, dan manajemen kunci kriptografi.

Sementara itu, NIST (*National Institute of Standards and Technology*) menetapkan standar kebutuhan fungsional keamanan melalui FIPS-200 yang berisi 17 area yang berkaitan dengan keamanan. Ketujuh belas area tersebut adalah sebagai berikut:

1. Kendali akses (*access control*)
2. Kesadaran dan pelatihan (*awareness and training*)
3. Audit dan akuntabilitas
4. Sertifikasi, akreditasi, dan penilaian keamanan
5. Pengelolaan konfigurasi (*configuration management*)
6. Rencana *contingency*
7. Identifikasi dan autentikasi
8. Tanggapan insiden
9. Pemeliharaan
10. Perlindungan media
11. Perlindungan fisik dan lingkungan
12. Perencanaan
13. Keamanan personil
14. Penilaian risiko
15. Akuisisi sistem dan layanan
16. Perlindungan sistem dan komunikasi
17. Integritas sistem dan informasi

Daftar periksa untuk tahap inisiasi/perencanaan proyek disusun untuk mencakup kegiatan perencanaan pengembangan, penggunaan *tools* yang telah ditetapkan, pengamanan lingkungan pengembangan, identifikasi kasus penyalahgunaan (*abuse cases*), serta penentuan kebutuhan dan persyaratan keamanan.

Tabel 4.1. Daftar Periksa untuk Tahap Inisiasi/Perencanaan Proyek

No.	Item	Pertanyaan	Applied	Done
A. Perencanaan Pengembangan				
1.	Penilaian risiko awal	Apakah penilaian risiko awal sudah dibuat?	☐	☐
2.	Perhitungan aspek keamanan	Apakah aspek keamanan sudah dimasukkan dalam perhitungan biaya dan waktu penyelesaian proyek?	☐	☐
3.	Penunjukan tim dan penanggung jawab	Apakah sudah dibuat SK untuk penunjukan tim pengembangan *software* yang aman berikut dengan penanggung jawabnya?	☐	☐
4.	Perjanjian kerahasiaan	Apakah setiap anggota tim yang terlibat pengembangan sudah menandatangani perjanjian kerahasiaan/NDA (*Non-Disclosure Agreement*)?	☐	☐
5.	*Roles and responsibilities*	Apakah spesifikasi peran dan tanggungjawab keamanan dari tiap orang atau bagian organisasi dalam tim Secure SDLC sudah dibuat dan didokumentasikan, serta diintegrasikan dengan peran mereka dalam pengembangan?	☐	☐
6.	Perencanaan aturan akses oleh individu	Apakah aturan perilaku dari individu yang mengakses sistem sudah dibangun, didokumentasikan, di-*update* secara periodik, dan diimplementasikan?	☐	☐

No.	Item	Pertanyaan	Applied	Done
7.	Rencana pemantauan kepatuhan	Apakah rencana untuk mengetahui/memonitor kepatuhan (*compliance*) dan kesehatan/kelancaran proses Secure SDLC sudah dibuat?	☐	☐
B. Pemakaian *Tools* yang yang Ditetapkan				
1.	Identifikasi *Security tools*	Apakah *tools* keamanan yang dilibatkan dalam pengembangan sudah diidentifikasi?	☐	☐
2.	Kepatuhan pemakaian *tools*	Apakah pengembangan aplikasi sudah menggunakan *tools* sesuai dengan yang ditetapkan?	☐	☐
3.	*Setting tools*	Apakah *setting* dari *tools* tersebut seperti opsi *compiler* dan *warning* sesuai dengan yang ditetapkan?	☐	☐
4.	*Updated tools*	Apakah *tools* (termasuk *compiler* dan *interpreter*) yang digunakan merupakan versi terbaru dan kompatibel dengan versi sebelumnya?	☐	☐
5.	Keamanan pengunduhan	Apakah *development tools* dan *build tools* diunduh dari *site* yang terpercaya?	☐	☐
6.	Integritas *tools*	Apakah *tools* diverifikasi integritasnya saat pengunduhan dan secara berkala?	☐	☐
7.	Versi terbaru	Apakah tools yang digunakan merupakan versi yang terbaru di mana cacat keamanan pada versi sebelumnya telah diperbaiki?	☐	☐

No.	Item	Pertanyaan	Applied	Done
8.	Pemantauan *tools* dan *log*nya	Apakah *tools* secara kontinyu dipantau berikut dengan *log*-nya untuk antisipasi adanya isu keamanan?	☐	☐
C. Pengamanan Lingkungan Pengembangan				
C.1. Pengamanan Fisik				
1.	Pembatasan akses fisik	Apakah sudah dilakukan pembatasan akses fisik terhadap sistem informasi, peralatan, dan lingkungan pengoperasian?	☐	☐
2.	Perlindungan akses fisik	Apakah bangunan fisik dan infrastruktur sudah dilindungi dan disediakan utilitas pendukungnnya?	☐	☐
3.	Kendali lingkungan fisik	Apakah telah ada kendali/kontrol yang cukup terhadap lingkungan fisik/fasilitas?	☐	☐
C.2. Pengamanan Lojik				
1.	*Submitting code and running builds*	Apakah dilakukan pembatasan terhadap siapa yang dapat men-*submit* kode dan menjalankan *builds*?	☐	☐
2.	Pemakaian VMs dan *containers*.	Apakah lingkungan pengujian menggunakan *containers* dan VMs?	☐	☐
3.	Penonaktifan elemen yang tidak dibutuhkan	Apakah akun, program, *service*, dan *port* yang tidak diperlukan sudah dinonaktifkan?	☐	☐
4.	*Security Patches*	Apakah sistem di-*update* secara regular dengan *security patches*?	☐	☐

No.	Item	Pertanyaan	Applied	Done
5.	*Unneccessary services*	Apakah servis-servis yang tidak diperlukan sudah dinonaktifkan?	☐	☐
6.	*Log* dan monitor	Apakah operasi dan *alerts* dari komponen pada lingkungan pengembangan dicatat dan dipantau secara berkesinambungan?	☐	☐
7.	Manajemen memori	Apakah sudah dilakukan manajemen memori untuk mengalokasikan bagian memori secara dinamis ke program sesuai permintaan, dan membebaskannya untuk digunakan kembali saat tidak lagi diperlukan ? (PABW 3.10 Manajemen Memori)	☐	☐
C.3. Pengamanan Fisik dan/atau Lojik				
1.	Akses ke lingkungan pengembangan	Apakah dilakukan pembatasan akses atau koneksi untuk dapat masuk ke dalam lingkungan pengembangan?	☐	☐
2.	Penggunaan *Firewalls* / WAF	Apakah *firewalls* atau *Web Application Firewalls* (WAF) sudah digunakan? (PABW 3.14.12 *Server Side Request Forgery* (SSRF))	☐	☐
3.	Pemisahan lingkungan pengembangan	Apakah sudah dilakukan pemisahan atau segmentasi antara lingkungan pengembangan (*development*) dengan lingkungan operasional (*production*)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
D. Identifikasi Kasus Penyalahgunaan (*Abuse Cases*)				
1.	Indentifikasi *abuse cases*	Apakah kasus penyalahgunaan (*abuse cases*) sudah diidentifikasi?	☐	☐
2.	*Use case* baru untuk mitigasi *abuse cases*	Apakah *use case* baru telah dibuat untuk memitigasi *abuse cases* dan memperkuat sistem?	☐	☐
3.	Identifikasi *misuse cases*	Apakah kasus pengguna baik-baik yang menjadi korban *phising* (*misuse case*) sudah diidentifikasi?	☐	☐
4.	*Use case* baru untuk mitigasi *misuse cases*	Apakah *use case* baru telah dibuat untuk memitigasi *misuse cases* dan memperkuat sistem?	☐	☐
E. Penentuan Kebutuhan Keamanan				
1.	Pengumpulan kebutuhan	Apakah kebutuhan keamanan sudah dikumpulkan dari semua pemangku kepentingan (*stakeholders*)?	☐	☐
2.	Standard dan regulasi internal	Apakah kebutuhan keamanan sudah mengakomodasi standard dan regulasi internal?	☐	☐
3.	Standard dan regulasi eksternal	Apakah kebutuhan keamanan sudah mengakomodasi standard dan regulasi eksternal?	☐	☐
4.	Insiden dan kegagalan sebelumnya	Apakah kebutuhan keamanan sudah mempertimbangkan insiden dan kegagalan keamanan sebelumnya?	☐	☐
5.	Risiko umum keamanan aplikasi web	Apakah kebutuhan sudah memasukkan 10 risiko umum keamanan aplikasi web dari OWASP?	☐	☐

No.	Item	Pertanyaan	Applied	Done
6.	Kebutuhan fitur-fitur keamanan	Apakah sudah dispesifikasikan fitur-fitur keamanan seperti *backup*, pemulihan/*recovery*, kriptografi, *logging*, autentikasi, dan otorisasi?	☐	☐
7.	Dokumentasi kebutuhan keamanan	Apakah kebutuhan keamanan tersebut sudah didokumentasikan/diarsipkan, serta dilakukan pencatatan mana yang diterima dan mana yang ditolak?	☐	☐
8.	Kepatuhan terhadap kebijakan pengarsipan	Apakah pengarsipan kebutuhan keamanan sudah sesuai dengan kebijakan organisasi (misalnya format, masa retensi, di mana harus disimpan, dan lain-lain)?	☐	☐
9.	Kecepatan pemulihan	Apakah sudah ditentukan mengenai seberapa cepat aplikasi dapat pulih/*recover* dari serangan keamanan?	☐	☐
10.	Prioritas kebutuhan	Apakah sudah ada prioritas mengenai berbagai kebutuhan keamanan yang akan diakomodasi?	☐	☐
11.	Reviu dan *update* kebutuhan	Apakah kebutuhan keamanan sudah secara teratur direviu dan diperbaharui (misalnya satu tahun sekali) sebagai refleksi dari perubahan kebutuhan fungsional dan lingkungan, usulan pihak internal dan eksternal, serta insiden keamanan?	☐	☐

BAB V
RANCANG BANGUN

5.1. Pendefinisian Standar Kriptografi

Kriptografi merupakan mekanisme perlindungan data yang bertujuan mencegah pengungkapan atau pengubahan data rahasia. Penggunaan kriptografi sangat penting untuk melindungi keamanan dan kerahasiaan data sensitif, baik saat data ditransmisikan maupun disimpan. Kriptografi dapat dilakukan melalui mekanisme berikut:

1. Mekanisme terbuka:
    - Enkripsi, yaitu proses untuk mengubah data berupa teks yang mudah dibaca (*clear text*) menjadi teks tersandi dengan menggunakan algoritma tertentu.
    - *Hashing,* yaitu proses menghasilkan kode dengan panjang tetap yang digunakan untuk mewakili kata, pesan atau data yang panjangnya bervariasi melalui penggunaan rumus matematika.
2. Mekanisme terselubung:
    - Steganografi, yaitu teknik menyembunyikan pesan dengan suatu cara sehingga tidak ada yang mengetahui atau menyadari bahwa ada suatu pesan rahasia, selain si pengirim dan si penerima;
    - *Digital watermarking,* yaitu teknik menyembunyikan informasi digital dalam sinyal pembawa, yang digunakan untuk memverifikasi keaslian, integritas, atau identitas pemilik dari sinyal pembawa tersebut.

5.2. Pendefinisian Rancangan Keamanan

Rancangan keamanan mengacu pada teknik dan metode yang digunakan untuk menerapkan mekanisme keamanan dalam sistem. Perancangan keamanan harus mempertimbangkan aspek-aspek inti, yaitu: kerahasiaan (*confidentiality*), integritas (*integrity*), ketersediaan (*availability*), autentikasi (*authentication*), otorisasi (*authorization*), dan kenirsangkalan (*non-repudiation*).

Untuk mengatasi potensi cacat atau kekurangan dalam perancangan keamanan, pengembang dapat menerapkan pemodelan ancaman, serta melakukan *review* terhadap arsitektur dan rancangan. Rancangan keamanan juga harus mencakup strategi enkripsi yang tepat untuk melindungi data sensitif, baik saat disimpan maupun ditransmisikan. Dalam hal autentikasi dan otorisasi, sistem harus menerapkan pendekatan yang telah distandardisasi untuk pengelolaan identitas

dan akses. Standarisasi ini memberikan konsistensi antar komponen serta pedoman yang jelas untuk melakukan verifikasi dan kontrol keamanan. *Log files* juga perlu dirancang untuk mengumpulkan informasi yang relevan terkait keamanan, namun dengan batasan agar tidak merekam terlalu banyak data. Pedoman dari OWASP (Open Web Application Security Project) dapat digunakan sebagai referensi dalam merancang dan mengimplementasikan *log files* secara efisien dan aman.

5.3. Pencegahan Cacat Umum Rancangan Keamanan

Kelemahan keamanan dalam rancangan sistem harus dihindari untuk memastikan perlindungan yang kuat terhadap potensi serangan. IEEE (Institute of Electrical and Electronics Engineers) telah mengidentifikasi 10 cacat rancangan keamanan yang paling umum, serta teknik-teknik yang dapat digunakan untuk menghindarinya. Tabel 5.1 merinci sepuluh cacat rancangan keamanan tersebut beserta solusi yang dapat diambil untuk mencegah terjadinya kelemahan pada sistem.

Tabel 5.1. Pencegahan Cacat Umum Rancangan Keamanan

No.	Item	Solusi	Keharusan/Larangan
1	Salah asumsi mengenai hak/ kepercayaan.	Memberikan hak/ kepercayaan, tetapi tidak boleh berasumsi tanpa dasar yang jelas.	• Memastikan bahwa semua data dari klien yang tidak dapat dipercaya telah divalidasi. • Mengasumsikan bahwa setiap data berpotensi diretas. • Melarang otorisasi, kendali akses, implementasi aturan keamanan, dan penggunaan data sensitif dijalankan pada kode yang dieksekusi di sisi klien.
2	Mekanisme otentikasi yang dapat dilewati atau dirusak.	Menggunakan mekanisme otentikasi yang tidak dapat dilewati atau dirusak.	• Mencegah pengguna mengubah profil atau identitas tanpa re-otentikasi setelah otentikasi awal. • Mempertimbangkan kekuatan autentikasi berdasarkan jenis pengguna. • Mengimplementasikan *timeouts*.

No.	Item	Solusi	Keharusan/Larangan
			• Mempertimbangkan tiga faktor autentikasi: apa yang diketahui, apa yang dimiliki, dan siapa yang melakukan autentikasi. • Melarang pembagian informasi sensitif seperti alamat IP dan MAC address. • Melarang penggunaan token yang mudah diprediksi atau ditebak.
3	Mengabaikan otorisasi setelah otentikasi.	Melakukan otorisasi setelah proses otentikasi.	• Melakukan otorisasi sebagai pemeriksaan yang eksplisit. • Menggunakan kembali fungsi atau modul umum untuk otorisasi. • Mengingat bahwa otorisasi bergantung pada konteks permintaan (misalnya, lokasi atau waktu), bukan hanya pada hak akses yang diberikan.
4	Kurangnya pemisahan yang jelas antara data dan instruksi kendali.	Memisahkan data dan instruksi kendali dengan jelas, dan tidak memproses instruksi kendali dari sumber yang tidak dapat dipercaya.	• Memanfaatkan kemampuan sistem operasi atau perangkat keras untuk memisahkan kode program dan data. • Mengaktifkan *flags* atau opsi keamanan pada *compiler/linker*. • Mengidentifikasi metode atau *endpoints* yang menggunakan tipe data terstruktur. • Melarang pencampuran data dan instruksi kendali dalam satu entitas. • Melarang penggunaan API yang rentan terhadap injeksi seperti XSS, SQL *injection*, dan *shell injection*.

No.	Item	Solusi	Keharusan/Larangan
5	Tidak mem-validasi semua data secara eksplisit.	Mendefinisikan metode yang memastikan semua data divalidasi secara eksplisit.	• Memastikan bahwa validasi data dilakukan secara menyeluruh dan komprehensif. • Menggunakan *whitelisting* untuk validasi data. • Melarang penggunaan *blacklisting*. • Menggunakan mekanisme validasi terpusat dengan format data yang telah ditentukan. • Melakukan reviu keamanan terhadap skema validasi. • Melarang asumsi terkait validitas data tanpa validasi yang tepat.
6	Penggunaan kriptografi yang tidak tepat.	Menggunakan kriptografi yang benar dan sesuai standar.	• Menggunakan algoritma dan pustaka kriptografi yang sudah distandardisasi. • Mengimplementasikan kriptografi secara terpusat. • Mendapatkan bantuan dari ahli kriptografi. • Melarang pembuatan sistem kriptografi sendiri karena terlalu rumit dan rawan kesalahan. • Melarang pengabaian isu manajemen kunci. • Melarang implementasi kriptografi yang tidak acak.
7	Kegagalan mengidentifikasi data sensitif dan cara menanganinya.	Mengidentifikasi data sensitif dan menangani sesuai dengan standar keamanan.	• Mengidentifikasi lokasi penyimpanan dan pemrosesan data sensitif. • Mengklasifikasikan data berdasarkan tingkat keamanannya. • Menerapkan pengendalian data seperti perlindungan file, memori, dan basis data.

No.	Item	Solusi	Keharusan/Larangan
			• Mempertimbangkan bahwa sensitivitas data dapat bergantung pada konteks. • Melarang pengabaian batasan/lingkup kepercayaan (*trust boundaries*).
8	Kegagalan mempertimbangkan aspek pengguna.	Mempertimbangkan aspek pengguna dalam setiap tahap pengembangan.	• Mempertimbangkan latar belakang budaya, pengalaman, dan persepsi pengguna. • Mengamankan segala sesuatu secara default. • Melarang anggapan bahwa keamanan adalah fitur tambahan. • Mengimplementasikan keamanan yang memadai dan sesuai kebutuhan. • Melarang asumsi bahwa pengguna peduli atau sadar akan keamanan. • Melarang penyerahan keputusan keamanan kepada pengguna.
9	Tidak memahami dampak integrasi komponen eksternal.	Memahami bagaimana integrasi komponen eksternal memengaruhi titik serangan.	• Melakukan pengujian keamanan untuk setiap komponen eksternal. • Memasukkan komponen eksternal ke dalam proses reviu keamanan. • Mengisolasi komponen eksternal dari sistem inti. • Memantau perkembangan informasi keamanan terkait komponen eksternal yang digunakan. • Menghindari asumsi bahwa *open-source* selalu aman. • Melarang penggunaan komponen eksternal tanpa kontrol keamanan yang jelas.

No.	Item	Solusi	Keharusan/Larangan
10	Kerapuhan terhadap perubahan objek dan aktor di masa depan.	Merancang sistem yang fleksibel terhadap perubahan objek dan aktor di masa depan.	• Merancang sistem dengan mempertimbangkan perubahan di masa depan. • Mengimplementasikan pembaruan keamanan secara berkala. • Menggunakan *code signing* dan perlindungan kode. • Menyusun rencana pemulihan jika terjadi pelanggaran keamanan. • Melarang pengabaian keamanan yang lemah atau rapuh. • Melarang peremehan terhadap administrasi dan operasi sistem. • Menyadari bahwa menjaga kerahasiaan adalah tugas yang sulit. • Menyadari bahwa semua sistem kriptografi berpotensi untuk ditembus.

5.4. Pemodelan Ancaman (*Threat Modelling*)

Pemodelan ancaman adalah metode untuk menciptakan abstraksi dari perangkat lunak dengan tujuan mengidentifikasi motif dan kemampuan penyerang serta menginventarisasi potensi ancaman yang harus dimitigasi. Pemodelan ini memberikan pandangan tentang aplikasi dan lingkungannya dari perspektif keamanan. Dalam model *Touchpoints*, pemodelan ancaman dikenal sebagai analisis risiko arsitektur (*architectural risk analysis*). Tujuan utama dari pemodelan ancaman adalah untuk meningkatkan keamanan aplikasi melalui identifikasi ancaman, dan mendefinisikan tindakan untuk mencegah dan mengurangi dampak ancaman. Selain itu, pemodelan ini juga berfungsi untuk mencegah dan mendeteksi cacat desain (*design flaws*).

Pemodelan ancaman dilakukan dengan memetakan setiap komponen sistem terhadap ancaman STRIDE *(Spoofing, Tampering, Repudiaton, Information Disclosure, Denial of Service, Elevation of Privilege*), yaitu:

1. *Spoofing identity* (pemalsuan identitas)
   Melanggar aspek autentikasi dengan memungkinkan penyerang berpura-pura menjadi pengguna sah atau membuat *server* palsu menyerupai yang asli.
2. *Tampering with data* (perusakan data)
   Ancaman ini melanggar integritas data melalui modifikasi data untuk tujuan jahat, seperti perubahan yang tidak sah.
3. *Repudiation* (penyangkalan)
   Ancaman ini melanggar aspek kenirsangkalan di mana pihak yang melakukan tindakan menyangkalnya, sementara pihak lain tidak dapat membuktikan kebenaran atau keberadaan aksi tersebut.
4. *Information disclosure* (pengungkapan informasi)
   Ancaman ini melanggar kerahasiaan dengan membocorkan informasi kepada pihak yang tidak berwenang.
5. *Denial of service* (kegagalan layanan)
   Ancaman ini melanggar ketersediaan layanan dengan menyebabkan layanan tidak tersedia atau tidak dapat digunakan oleh pengguna sah.
6. *Elevation of privilege* (pengubahan/peningkatan hak akses)
   Ancaman ini melanggar otorisasi, di mana pengguna yang tidak berwenang memperoleh hak akses yang lebih tinggi dan dapat merusak atau mengambil alih sistem.

Pemodelan ancaman juga mencakup pemilihan fitur keamanan yang tepat untuk mengurangi risiko ancaman, seperti penggunaan kriptografi, autentikasi, dan otorisasi. Tim pengembang perlu merancang tindakan untuk mengurangi wilayah serangan (*attack surface*), yang merupakan titik-titik lemah di mana penyerang dapat mencoba mengeksploitasi sistem.

Daftar periksa untuk tahap rancang bangun disusun untuk mencakup kegiatan pendefinisian standar kriptografi, pendefinisian rancangan keamanan, pencegahan cacat umum dalam rancangan keamanan, dan pemodelan ancaman, sebagaimana ditunjukkan oleh Tabel 5.2 berikut.

Tabel 5.2. Daftar Periksa untuk Tahap Rancang Bangun

No.	Item	Pertanyaan	Applied	Done
A. Pendefinisian Standar Kriptografi				
1	Enkripsi data sensitif	Apakah sudah ditentukan data kritikal/sensitif, seperti kata sandi dan informasi kesehatan pribadi, baik yang disimpan maupun yang dipertukarkan untuk dilindungi dari pengungkapan melalui enkripsi searah/dua arah?	☐	☐
2.	Keamanan kriptografi	Apakah standar enkripsi yang digunakan sudah cukup aman dan merupakan standar yang diterima banyak kalangan (tidak membuat sendiri), dengan menggunakan algoritma dan panjang kunci yang layak?	☐	☐
3.	Pustaka enkripsi	Apakah sistem sudah didesain untuk memungkinkan pustaka enkripsi (*encryption libraries*) dapat diganti dengan mudah jika sewaktu-waktu pustaka tersebut diretas?	☐	☐
4.	*Sufficient length and randomness*	Jika keamanan algoritma kriptografi tergantung kepada bilangan atau karakter acak, apakah sudah dipastikan bahwa panjang dan keacakannya cukup?	☐	☐
5.	Keberlakuan kunci kriptografi	Apakah kunci kriptografi tidak kedaluwarsa, ditarik (*revoked*) atau *compromised*?	☐	☐
6.	Perubahan periodik kunci	Apakah kunci kriptografi diperbaharui atau diganti secara periodik?	☐	☐

No.	Item	Pertanyaan	Applied	Done
7.	Pengelolaan kunci kriptografi	Apakah kunci kriptografi telah disimpan dengan aman dan di-*backup*?	☐	☐
B. Pendefinisian Desain Keamanan				
B.1 Pengelolaan Desain				
1	Akomodasi kebijakan keamanan	Apakah arsitektur dan desain dibuat untuk mengimplementasikan dan melaksanakan kebijakan keamanan?	☐	☐
2	Kesederhanaan desain	Apakah desain tetap dipertahankan agar tetap sederhana dan cukup kecil (*keep it simple*)?	☐	☐
3	Penanganan kebutuhan yang tidak diakomodasi	Jika ada kebutuhan keamanan yang tidak dapat diakomodasi dalam desain karena satu dan lain hal, apakah desain atau strategi penanganan risiko diubah atau dilakukan pencatatan khusus mengenai hal tersebut?	☐	☐
4	Asumsi keamanan	Apa saja asumsi dalam keamanan (misalnya semua pengguna mematuhi kebijakan keamanan, atau *vendor* sudah menyediakan perlindungan keamanan yang cukup)?	☐	☐
5	Kesalahan asumsi	Apa saja yang dapat membuat asumsi dalam keamanan salah (misalnya perubahan lingkungan, kurangnya kesiapan terhadap serangan)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
B.2. Kerahasiaan				
1	Strategi enkripsi	Apakah strategi enkripsi sudah dispesifikasikan baik terhadap data yang disimpan maupun data yang ditransimisikan?	☐	☐
2	Atribut dan pengelolaan enkripsi	Apakah algoritma, panjang kunci, manajemen kunci (termasuk penyimpanan, pertukaran, rotasi kunci) sudah cukup memadai?	☐	☐
3	Informasi non-sensitif pada file log	Apakah sudah dipastikan bahwa file log tidak menyimpan informasi sensitif dalam bentuk yang dapat dibaca manusia atau mudah diuraikan (misalnya pencatatan kesalahan pengetikan *password*)?	☐	☐
4	Rancangan antarmuka aman	Apakah antarmuka sudah dirancang secara aman, seperti *masking* informasi sensitif pada kata sandi, atau nomor kartu kredit, atau menggunakan *database view*?	☐	☐
5	Klasifikasi informasi	Apakah informasi telah diklasifikasikan menurut tingkat keamanannya?	☐	☐
6	Identifikasi dan penyimpanan informasi dikecualikan	Apakah sudah dilakukan identifikasi jenis dan penyimpanan salinan informasi yang dikecualikan? (PABW 3.9 Proteksi Data)	☐	☐
7	Pelindungan informasi yang	Apakah informasi yang dikecualikan tersebut, ketika disimpan sementara dalam	☐	☐

No.	Item	Pertanyaan	Applied	Done
	dikecualikan dari akses tidak sah	aplikasi sudah dilindungi dari akses yang tidak sah? (PABW 3.9 Proteksi Data)		
8	Pertukaran, penghapusan, dan audit informasi	Apakah proses pertukaran dan penghapusan informasi yang dikecualikan sudah dilaksanakan secara aman dan diaudit/ dievaluasi? (PABW 3.9 Proteksi Data)	☐	☐
9	Minimalisasi parameter *request*	Apakah parameter yang dibutuhkan untuk *request* ke *server* sudah diminimalkan? (PABW 3.9 Proteksi Data)	☐	☐
10	Penyimpanan data yang aman	Apakah sudah dipastikan bahwa data disimpan dengan aman? (PABW 3.9 Proteksi Data)	☐	☐
11	Metode aman untuk hapus dan ekspor data	Apakah sudah ditentukan metode yang aman untuk menghapus dan mengekspor data sesuai permintaan pengguna? (PABW 3.9 Proteksi Data)	☐	☐
12	Pembersihan memori yang tidak terpakai	Apakah memori yang sudah tidak diperlukan telah dibersihkan? (PABW 3.9 Proteksi Data)	☐	☐
13	Penyimpanan data dan informasi	Apakah seluruh data dan informasi yang dikecualikan disimpan dalam fasilitas penyimpanan khusus pada sistem?	☐	☐

No.	Item	Pertanyaan	Applied	Done
14	Pembatasan pertukaran dengan *third party*	Apakah pertukaran data dan informasi yang dikecualikan dengan *third party* dibatasi? (PABM 3.7 Penanganan Penyimpan Data)	☐	☐
15	Penonaktifan *cache keyboard* saat *input*	Apakah *cache keyboard* dinonaktifkan pada saat memasukkan data dan informasi yang dikecualikan? (PABM 3.8 Kerawanan Kontrol Privasi)	☐	☐
16	Pelindungan *interprocess communication*	Apakah informasi yang dikecualikan dilindungi saat terjadi *interprocess communication*? (PABM 3.7 Penanganan Penyimpan Data)	☐	☐
17	Pelindungan *input* melalui antarmuka	Apakah data dan informasi yang dikecualikan yang dimasukkan melalui antarmuka pengguna sudah dilindungi? (PABM 3.7 Penanganan Penyimpan Data)	☐	☐
18	Data sensitif via *custom URL* dan *interprocess communication*	Apakah sudah dihindari pengiriman informasi/fungsionalitas sensitif melalui skema *custom uniform resource locator* dan fasilitas *interprocess communication*? (PABM 3.9 Kerawanan Komunikasi)	☐	☐

No.	Item	Pertanyaan	Applied	Done
19	Menyembunyikan informasi *web server*	Apakah informasi terkait *server* sudah disembunyikan? (PABW 2.3.6 Menyembunyikan Informasi Versi *Web Server*)	☐	☐
B.3. Integritas				
1	Pembedaan *input* baik dan buruk	Apakah *software* dirancang agar dapat membedakan antara input yang baik dan yang buruk (misalnya *filter input* serta perlindungan akses dari konten *script* dan injeksi *script*)?	☐	☐
2	Pembedaan aplikasi resmi dengan yang bukan.	Apakah sistem dirancang agar dapat membedakan permintaan (*request*) antara aplikasi yang terlegitimasi dengan aplikasi tipuan (*rogue*)?	☐	☐
3	DB *referential integrity*	Apakah rancangan basis data sudah menerapkan *referential integrity*?	☐	☐
4	Integritas data log	Apakah data log diproteksi (misalnya menggunakan *hash*) dan menggunakan *timestamp* sistem (misalnya menggunakan fungsi getDate() pada SQL) alih-alih ditentukan pengguna?	☐	☐
5	Deteksi modifikasi data	Apakah sistem dirancang agar dapat mendeteksi modifikasi data yang tidak seharusnya (misalnya dengan *hash function* atau tanda tangan elektronik tersertifikasi)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
6	Deteksi modifikasi perangkat	Apakah ada pencegahan jika aplikasi berjalan pada perangkat yang telah dilakukan modifikasi yang tidak sah? (PABM 3.12 Kerawanan Proteksi *Binary*)	☐	☐
7	Deteksi perubahan di memori	Apakah aplikasi dapat mendeteksi perubahan kode dan data di ruang memori? (PABM 3.16 Penanganan Memori)	☐	☐
8	Penerapan fungsi *device binding*	Apakah aplikasi menerapkan fungsi *device binding* dengan menggunakan property unik pada perangkat? (PABM 3.12 Kerawanan Proteksi *Binary*)	☐	☐
9	Pelindungan *file* dan *library*	Apakah seluruh *file* dan *library* pada aplikasi dilindungi?	☐	☐
10	Pembaruan dan *patching library*	Apakah *library* yang digunakan sudah diperbaharui dengan versi yang terkini? (PABM 2.2.2 Pembaruan dan *Patching* IDE)	☐	☐
B.4. Ketersediaan (*Availability*)				
1	*Resource locking management*	Apakah sudah diidentifikasi bagaimana mekanisme *resource locking* pada basis data, terutama untuk data yang diakses secara bersamaan?	☐	☐

No.	Item	Pertanyaan	Applied	Done
2	Dukungan ketersediaan	Untuk sistem dengan kebutuhan ketersediaan tinggi, apakah telah dirancang strategi untuk menjamin data dan informasi dapat selalu diakses seperti replikasi, pengalihan ke sistem redundan, dan skalabilitas?	☐	☐
3	File log untuk forensik	Apakah file log sudah didefinisikan agar analisis forensik dapat dilakukan ketika terjadi gangguan?	☐	☐
4	Kemampuan file log	Apakah file log untuk aplikasi, sistem dan keamanan telah didesain agar dapat menggambarkan perilaku aplikasi, bagaimana penggunaannya sepanjang waktu, serta dapat membedakan antara perilaku pengguna yang baik dan yang memberikan ancaman?	☐	☐
5	Efisiensi file log	Apakah file log didesain secara efisien agar dapat mengumpulkan/menyediakan berbagai informasi penting, tetapi tidak merekam terlalu banyak data?	☐	☐
6	*Backup and recovery*	Apakah sudah dirancang proses pencadangan dan pemulihan data dan sistem?	☐	☐
7	Jaminan *uptime*	Apakah telah ditetapkan target jaminan *uptime* sistem (misalnya 99%)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
B.5. Autentikasi				
1	Kata sandi	Jika sebuah situs web atau operasi memerlukan kontrol akses, apakah sudah digunakan manajemen kata sandi untuk proses autentikasi?	☐	☐
2	Verifikasi kata sandi di *server*	Apakah verifikasi kata sandi dilakukan di sisi *server*?	☐	☐
3	Standarisasi untuk *identity and access management*	Apakah sistem menggunakan pendekatan/metode yang terstandarisasi untuk pengelolaan identitas dan akses dalam autentikasi dan otorisasi (misalnya OAuth 2.0, autentikasi multi-faktor, atau federasi identitas dengan *single sign-on*)?	☐	☐
4	*Password* yang kuat	Apakah kata sandi sudah cukup kuat dengan menetapkan: - Jumlah karakter minimal 12; - Kombinasi jenis karakter mininal terdiri atas huruf besar, huruf kecil, simbol dan angka;	☐	☐
5	*Multifactor authenticatio n*	Apakah autentikasi dirancang dengan mengakomodasi fitur *multifactor authentication (MFA)*?	☐	☐
6	Jumlah *unsuccessful login*	Apakah sudah diatur jumlah maksimum kesalahan dalam pemasukan kata sandi maksimal 3 kali percobaan?	☐	☐
7	Masa berlaku *password*	Apakah aplikasi sudah mengakomodasi periode	☐	☐

No.	Item	Pertanyaan	Applied	Done
	dan nonaktif akun	kedaluwarsa *password* (misalnya 6 bulan) tanpa ada perulangan kata sandi dan mengakomodasi penonaktifan akun?		
8	Pemulihan kata sandi	Apakah sudah diatur mekanisme pemulihan kata sandi?	☐	☐
9	Penyimpanan kata sandi dengan kriptografi	Apakah sudah dipastikan bahwa aplikasi tidak menyimpan data *credential* secara *plaintext* (sebaiknya menggunakan *one-way hash* dengan *salt*)? (PABW 3.13 Kriptografi)	☐	☐
10	Jalur komunikasi aman	Apakah jalur komunikasi yang digunakan untuk proses autentikasi sudah diamankan? (PABW 2.3.2 Menggunakan HTTPS)	☐	☐
11	Penggunaan HTTPS pad WebView	Apakah digunakan protokol *hypertext transfer protocol secure* pada WebView? (PABM 3.13 Kerawanan WebView)	☐	☐
12	Autentikasi data	Apakah autentikasi data ikut dipertimbangkan dalam perancangan (misalnya melalui mekanisme verifikasi, validasi, atau *hash function*)?	☐	☐
13	Autentikasi pada *remote endpoint*	Apakah autentikasi diterapkan pada *remote endpoint* terhadap aplikasi yang menyediakan akses pengguna untuk layanan jarak jauh?	☐	☐

No.	Item	Pertanyaan	Applied	Done
		(PABM 3.3 Kerawanan Otentikasi)		
14	Memastikan pengguna yang sesungguhnya	Apakah sudah dipastikan bahwa seorang pengguna tidak dapat berpura-pura menjadi pengguna lainnya (misalnya melalui MFA atau pengelolaan sesi)?	☐	☐
B.6. Otorisasi				
1	*Least Privilege*	Apakah manajemen pengguna sudah menerapkan *The Principle of Least Privilege*?	☐	☐
2	*Never alone and segregation of duties*	Apakah manajemen akses pengguna sudah menerapkan prinsip '*never alone*' dan '*segregation of duties*'?	☐	☐
3	Tempat penentuan akses	Apakah terdapat kesepakatan mengenai di mana hak akses data didefinisikan (*database* vs *server aplikasi*)?	☐	☐
4	Pembatasan kontrol akses	Apakah sudah dilakukan penentuan pembatasan akses terhadap pengguna yang sah, proses yang berjalan atas nama pengguna, atau perangkat (termasuk sistem informasi lain) serta jenis transaksi dan fungsi yang melibatkan pengguna? (PABW 2.3.5: Melakukan Kontrol Akses, 3.14.1 Broken Access Control)	☐ ☐	☐
5	Skema kendali akses atau *access*	Apakah skema kendali akses (*access control schemes*) atau daftar kendali akses (*access control*	☐	☐

No.	Item	Pertanyaan	Applied	Done
	control lists (ACLs)	*lists*) sudah didefinisikan (misalnya mengikuti skema *role-based*, *attribute-based, mandatory*, atau *discretionary*) sesuai dengan karakteristik project?		
6	Kebutuhan dan klasifikasi informasi	Apakah otorisasi dirancang dengan hak akses yang sesuai dengan kebutuhan dan klasifikasi informasi?	☐	☐
7	*No conflicting roles*	Apakah sudah dipastikan tidak ada *conflicting roles* yang melanggar prinsip pemisahan tugas (misalnya pengguna tidak dapat berperan sebagai pemohon dan verifikator sekaligus)?	☐	☐
8	Peringatan serangan otomatis	Apakah sudah diatur peringatan terhadap bahaya serangan otomatis apabila terjadi akses yang bersamaan atau akses yang terus-menerus pada fungsi/objek tertentu? (PABW 3.2 *Logging*)	☐	☐
7	Antarmuka pada administrator	Apakah antarmuka pada sisi administrator sudah diamankan?	☐	☐
8	Verifikasi token saat akses data	Apakah kebenaran token diverifikasi ketika mengakses data dan informasi yang dikecualikan? (PABW 3.11 API)	☐	☐
9	Otorisasi pada *remote endpoint*	Apakah dilakukan otorisasi pada *remote endpoint*?	☐	☐

No.	Item	Pertanyaan	Applied	Done
		(PABM 3.3 Kerawanan Otentikasi)		
10	Permintaan akses seperlunya	Apakah sudah dipastikan bahwa aplikasi meminta akses hanya terhadap sumber daya yang diperlukan? (PABM 3.11 Kerawanan Konfigurasi Keamanan)	☐	☐
B.7. Kenirsangkalan (*Non-repudiation*)				
1	Jaminan kenirsangkalan	Apakah sistem dirancang untuk menjamin aspek kenirsangkalan (misalnya dengan tanda tangan elektronik, *timestamping*, atau *audit trail* yang mencatat aktivitas sistem)?	☐	☐
2	Sinkronisasi waktu	Apakah waktu *server* atau perangkat lainnya disinkronisasi?	☐	☐
3	Sertifikat elektronik	Jika transaksi atau dokumen membutuhkan pengesahan, apakah kenirsangkalan sudah dijamin oleh penyelenggara sertifikat elektronik melalui fitur sertifikat elektronik?	☐	☐
B.8. Manajemen Sesi				
1	Penggunaan pengendali sesi	Apakah sudah digunakan pengendali sesi untuk proses manajemen sesi? (PABW 3.4 Manajemen Sesi)	☐	☐
2	Pengendali sesi dari *framework*	Apakah digunakan pengendali sesi yang disediakan oleh *framework* atau fungsi bawaan aplikasi? (PABW 3.4 Manajemen Sesi)	☐	☐

No.	Item	Pertanyaan	Applied	Done
3	Pembuatan dan keacakan token	Apakah sudah diatur pembuatan dan keacakan token sesi yang dihasilkan oleh pengendali sesi? (PABW 3.4 Pastikan Token Sesi Cukup Acak)	☐	☐
4	Kondisi dan waktu habis sesi	Apakah kondisi dan jangka waktu habis / kedaluwarsa sesi sudah diatur? (PABW 3.4 Terapkan Batas Sesi)	☐	☐
5	Pencantuman dan validasi sesi	Apakah id sesi dicantumkan (*session inclusion*) dan divalidasi? (PABW 3.4 Manajemen Sesi)	☐	☐
6	Pelindungan lokasi dan pengiriman token	Apakah dilakukan pelindungan terhadap lokasi dan pengiriman token untuk sesi terautentikasi? (PABW 2.3.2 Menggunakan HTTPS)	☐	☐
7	Pelindungan duplikasi sesi dan persetujuan pengguna	Apakah dilakukan pelindungan terhadap duplikasi sesi dan mekanisme persetujuan pengguna? (PABW 3.4 Terapkan Batas Sesi)	☐	☐
8	*Session identifier* tanpa pengiriman kredensial	Apakah digunakan *session identifier* yang acak tanpa perlu mengirimkan kredensial pengguna apabila menggunakan *stateful* manajemen sesi? (PABM 4.6 Pengujian Keamanan Dinamis)	☐	☐
9	Token aman untuk	Apabila menggunakan autentikasi *stateless* berbasis token, apakah	☐	☐

No.	Item	Pertanyaan	Applied	Done
	autentikasi *stateless*	*server* menyediakan token yang telah ditandatangani menggunakan algoritma yang aman? (PABM 3.4 Penanganan Token Otentikasi)		
10	Pemutusan sesi saat pengguna *log out*	Apakah *remote endpoint* memutus sesi yang ada saat pengguna *log out*? (PABM 3.3 Kerawanan Otentikasi)	☐	☐
11	Sandi pada *remote endpoint*	Apakah diterapkan pengaturan sandi pada *remote endpoint*?	☐	☐
12	Pembatasan jumlah percobaan *log in*	Apakah jumlah percobaan *log in* pada *remote endpoint* dibatasi? (PABM 3.3 Kerawanan Otentikasi)	☐	☐
13	Masa berlaku dan kedaluwarsa token	Apakah sudah ditentukan masa berlaku sesi dan masa kedaluwarsa token pada *remote endpoint*? (PABM 3.3 Kerawanan Otentikasi)	☐	☐
14	ID sesi dan URL	Apakah ID sesi sulit ditebak dan tidak menggunakan parameter URL untuk menyimpan ID sesi?	☐	☐
15	*Setting cookie* secara aman	Apakah sudah dilakukan *setting cookie* yang aman (misalnya mengeset atribut “Secure”, “HttpOnly”, atau “SameSite”)?	☐	☐
16	Sesi baru setelah login	Apakah selalu dimulai sesi baru setelah login berhasil?	☐	☐

No.	Item	Pertanyaan	Applied	Done
17	Re-autentikasi pengguna	Apakah pengguna diautentikasi kembali setelah aktivitas tertentu atau sejumlah waktu tertentu?	☐	☐
B.9. Fungsi Kriptografi				
1	Kesesuaian kriptografi dengan peraturan	Apakah algoritma kriptografi, modul kriptografi, protokol kriptografi, dan kunci kriptografi yang digunakan sesuai dengan ketentuan peraturan perundang-undangan? (PABW 3.13. Kriptografi)	☐	☐
2	Autentikasi data yang dienkripsi	Apakah data yang dienkripsi diautentikasi untuk melindungi integritas data dan memastikan bahwa data berasal dari sumber yang sah?	☐	☐
3	Manajemen kunci kriptografi	Apakah sudah diterapkan manajemen kunci kriptografi (misalnya: pembuatan, distribusi, rotasi, penyimpanan, dan penghapusan kunci)? (PABW 3.3 Otentikasi dan Manajemen *Password*)	☐	☐
4	Generator angka acak kriptografi	Apakah angka acak yang dibuat menggunakan generator angka acak kriptografi? (PABW 3.3 Otentikasi dan Manajemen Password)	☐	☐
5	Menghindari *hardcoded*	Apakah sudah dihindari penggunaan kriptografi simetrik dengan *hardcoded key*?	☐	☐

No.	Item	Pertanyaan	Applied	Done
	key kriptografi	(PABM 3.1 Penanganan Data Kredensial)		
6	Menggunakan kriptografi yang teruji	Apakah metode kriptografi yang diimplementasikan sudah teruji sesuai kebutuhan? (PABM 3.6 Kerawanan Kriptografi)	☐	☐
7	Menghindari protokol dan algoritma kriptografi usang	Apakah sudah dihindari penggunaan protokol kriptografi atau algoritma kriptografi yang obsolet? (PABM 3.6 Kerawanan Kriptografi)	☐	☐
8	Menghindari kunci kriptografi sama	Apakah sudah dihindari penggunaan kunci kriptografi yang sama? (PABM 3.6 Kerawanan Kriptografi)	☐	☐
9	Menggunakan pembangkit kunci acak	Apakah sudah digunakan pembangkit kunci acak yang memenuhi kriteria keacakan kunci? (PABM 3.4 Penanganan Token Otentikasi)	☐	☐
B.10. Keamanan Komunikasi (Interkonektivitas)				
1	Keamanan dan enkripsi jalur komunikasi	Apakah jalur komunikasi, termasuk API, *web services, dan* token autentikasi sudah diamankan dan dienkripsi? (PABW 2.3.2 Menggunakan HTTPS)	☐	☐
2	Keamanan koneksi	Apakah sudah diatur koneksi masuk dan keluar yang aman dan terenkripsi dari sisi pengguna	☐	☐

No.	Item	Pertanyaan	Applied	Done
	masuk dan keluar	(misalnya implementasi *firewall*, WAF, proteksi DDoS, VPN/*Tunneling*)? (PABW 2.3.2 Menggunakan HTTPS, PABW 3.14.12 *Server Side Request Forgery* (SSRF)).		
3	Keamanan algoritma komunikasi	Apakah sudah diatur algoritma komunikasi dan alat pengujiannya? (PABW 2.3.3 Mengkonfigurasi TLS yang lemah di server, 2.3.4 Menggunakan Sertifikat HTTPS yang Valid dari *Certificate Authority* (CA) yang Terkemuka)	☐	☐
4	Aktivasi dan konfigurasi sertifikat elektronik	Apakah sudah diatur aktivasi dan konfigurasi sertifikat elektronik yang diterbitkan oleh penyelenggara sertifikasi elektronik? (PABW 2.3.4 Menggunakan Sertifikat HTTPS yang Valid dari *Certificate Authority* (CA) yang Terkemuka)	☐	☐
5	Penerapan SSL atau TLS	Apakah diterapkan *secure socket layer atau transport layer security* yang tidak obsolet secara konsisten? (PABM A.2 Analisis Dinamis)	☐	☐
6	Verifikasi sertifikat *remote end point*	Apakah sertifikat *remote endpoint* diverifikasi? (PABM A.2 Analisis Dinamis)	☐	☐

No.	Item	Pertanyaan	Applied	Done
B.11. Logika Bisnis				
1	Langkah dan waktu yang realistis	Apakah alur logika bisnis diproses dalam urutan langkah dan waktu yang realistis?	☐	☐
2	Batasan dan validasi logika bisnis	Apakah sudah dipastikan bahwa logika bisnis memiliki batasan dan validasi?	☐	☐
3	Membantu dalam kontrol antiotomatisasi	Apakah sudah dirancang fitur untuk membantu dalam kontrol antiotomatisasi (misalnya menggunakan CAPTCHA atau membatasi frekuensi permintaan dalam waktu tertentu)? (PABW 3.3 Otentikasi dan Manajemen *Password* pada *Brute Force*)	☐	☐
4	Peringatan pada aktivitas otomatis atau tidak biasa	Apakah sistem memberikan peringatan ketika terjadi serangan otomatis atau aktivitas yang tidak biasa? (PABW 3.3 Otentikasi dan Manajemen *Password* pada *Brute Force*)	☐	☐
B.12. Manajemen File				
1	Pengaturan jumlah dan ukuran *file*	Apakah jumlah *file* untuk setiap pengguna dan kuota ukuran *file* yang diunggah sudah dibatasi? (PABW 3.7 *File Management* dan 3.12 *File Upload*)	☐	☐

No.	Item	Pertanyaan	Applied	Done
2	Validasi konten *file*	Apakah dilakukan validasi file sesuai dengan tipe konten yang diharapkan? (PABW 3.7 *File Management* dan 3.12 *File Upload*)	☐	☐
3	Pelindungan metadata input dan *file*	Apakah dilakukan pelindungan terhadap *metadata input* dan *metadata file?* (PABW 3.12 *File Upload*)	☐	☐
4	Pemindaian *file* dari sumber tidak terpercaya	Apakah dilakukan pemindaian *file* yang diperoleh dari sumber yang tidak dipercaya? (PABW 3.7 *File Management*)	☐	☐
5	Konfigurasi *server* untuk *download file*	Apakah dilakukan konfigurasi *server* untuk mengunduh *file* sesuai ekstensi yang ditentukan? (PABW 3.7 *File Management* dan 3.12 *File Upload*)	☐	☐
6	Konfigurasi daftar direktori	Apakah daftar direktori pada *web server* sudah dinonaktifkan? (PABW 2.3.7 Menonaktifkan Daftar *File* dan Direktori)	☐	☐
7	Nama file via parameter eksternal	Apakah sudah dihindari untuk mencantumkan nama file yang disimpan di server web secara langsung melalui parameter eksternal?	☐	☐
8	Direktori tetap untuk file	Apakah digunakan direktori tetap untuk menangani nama file dan mengabaikan nama direktori dalam nama file?	☐	☐

No.	Item	Pertanyaan	Applied	Done
9	Pengelolaan izin akses file	Apakah izin akses file sudah dikelola dengan baik?	☐	☐
10	Validasi nama file	Apakah dilakukan validasi terhadap nama file?	☐	☐
B.13. Keamanan API dan Web Service				
1	Verifikasi keamanan URI API	Apakah *uniform resource identifier* API diverifikasi sehingga tidak menampilkan informasi yang berpotensi sebagai celah keamanan? (PABW 3.11 API)	☐	☐
2	Keputusan otorisasi entitas	Apakah dibuat keputusan otorisasi untuk memastikan entitas memiliki hak akses yang cukup untuk operasi tertentu? (PABW 3.11 API)	☐	☐
3	Penggunaan / penampilan RESTful HTTP	Apakah digunakan/ditampilkan metode RESTful *hypertext transfer protocol* apabila *input* pengguna dinyatakan valid? (PABW 3.11 API)	☐	☐
4	Validasi skema dan verifikasi	Apakah API/*Web Service* menggunakan validasi skema dan verifikasi sebelum menerima input? (PABW 3.11 API)	☐	☐
5	Menggunakan metode layanan berbasis web	Apakah sudah digunakan metode pelindungan layanan berbasis web (misalnya menggunakan protokol enkripsi HTTPS atau manajemen autentikasi)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
		(PABW 3.11 API)		
6	Penerapan kontrol antiotomatisasi	Apakah sudah diterapkan kontrol antiotomatisasi (misalnya menggunakan *captcha* atau membatasi trafik)? (PABW 3.11 API)	☐	☐
7	Serialisasi API yang aman	Apakah penggunaan serialisasi API (proses mengubah objek atau data menjadi format yang dapat ditransfer atau disimpan) diimplementasikan dengan aman? (PABM 3.15 Kerawanan Penggunaan API)	☐	☐
B.14. Ketahanan				
1	Deteksi *debugger*	Apakah aplikasi dapat mendeteksi dan merespons *debugger*?	☐	☐
2	Deteksi alat *reverse engineering*	Apakah aplikasi dapat mendeteksi dan merespons keberadaan perangkat *reverse engineering*? (PABM 3.12 Kerawanan Proteksi Binary)	☐	☐
3	Deteksi emulator	Apakah aplikasi dapat mencegah untuk berjalan dalam emulator? (PABM 3.12 Kerawanan Proteksi Binary)	☐	☐
4	Penerapan *obfuscation*	Apakah aplikasi menerapkan metode *obfuscation* (menggunakan teknik-teknik tertentu untuk membuat kode atau data menjadi sulit dimengerti atau dibaca oleh manusia)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
		(PABM 3.12 Kerawanan Proteksi Binary)		
B.15. Aspek Desain Keamanan Lainnya				
1	Modularisasi komponen keamanan	Apakah komponen keamanan dipisahkan dari komponen lainnya?	☐	☐
2	Kerentanan *database*	Apakah kerentanan *database* sudah diidentifikasi?	☐	☐
3	*Database protection*	Apakah telah dirancang perlindungan keamanan untuk basis data (misalnya dengan *polyinstantiaton*, enkripsi, normalisasi, serta pemanfaatan *trigger* dan *view*)?	☐	☐
4	Konfigurasi keamanan	Apakah telah dirancang konfigurasi/*setting* keamanan untuk berbagai komponen sistem seperti *web server*, *browser*, *compiler, database*, dan sistem operasi?	☐	☐
B.16. Reviu Desain				
1	Reviu desain dan arsitektur	Apakah desain dan arsitektur sudah di-reviu untuk mengidentifikasi cacat desain?	☐	☐
2	Reviu desain terhadap kebutuhan	Apakah desain/rancangan *software* sudah direviu untuk memastikan bahwa semua kebutuhan keamanan (*security requirements*) sudah diakomodasi?	☐	☐

No.	Item	Pertanyaan	Applied	Done
3	Reviu desain terhadap risiko	Apakah desain/rancangan software sudah direviu untuk memastikan bahwa risiko yang diidentifikasi sudah dipertimbangkan?	☐	☐
4	Perbaikan desain berdasarkan kebutuhan	Jika hasil reviu menyatakan ada kebutuhan keamanan yang belum diakomodasi, apakah desain telah diperbaiki?	☐	☐
5	Perbaikan desain berdasarkan risiko	Jika hasil reviu menyatakan ada risiko yang belum diakomodasi, apakah desain telah diperbaiki?	☐	☐
6	Pencatatan reviu dan tindakannya	Apakah hasil reviu berikut dengan tindak lanjutnya telah dicatat/didokumentasikan?	☐	☐
C. Pencegahan Cacat Umum Desain Keamanan				
1	10 cacat umum desain keamanan	Apakah 10 cacat umum desain keamanan menurut IEEE sudah dihindari (dibahas pada Subbab 5.3. Menghindari Cacat Umum Desain Keamanan)?	☐	☐
D. Pemodelan Ancaman (*Threat Modelling*)				
1	Deskripsi subjek	Apakah sudah ada deskripsi mengenai subjek yang akan dimodelkan?	☐	☐
2	Analisis risiko	Apakah sudah dilakukan analisis risiko terhadap karateristik *software*, ancaman, kerentanan, kemungkinan, dampak, dan perencanaan mitigasinya?	☐	☐

No.	Item	Pertanyaan	Applied	Done
3	Prioritas risiko	Apakah sudah dilakukan assessment terhadap berbagai risiko serta mengurutkannya berdasarkan dampak kerusakan (*severity*) dan kemungkinan (*likelihood*)?	☐	☐
4	Asumsi masa depan	Apakah model membuat asumsi/*trend* di masa depan di mana perkembangan ancaman keamanan berubah?	☐	☐
5	Potensi serangan / ancaman STRIDE	Apakah berbagai potensi ancaman STRIDE (*spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege*) terhadap sistem sudah diidentifikasi?	☐	☐
6	Penyampaian informasi ancaman	Apakah risiko atau potensi ancaman sudah dikomunikasikan kepada pemilik proses bisnis?	☐	☐
7	Mitigasi ancaman	Apakah sudah ada kendali/aksi yang dilakukan sebagai mitigasi dari setiap ancaman *spoofing, tampering, repudiation, information disclosure, denial of service, dan elevation of privilege*?	☐	☐
8	*Attack surface*	Apakah *attack surface* sudah diminimalkan?	☐	☐
9	Validasi dan verifikasi model	Apakah ada penjelasan mengenai cara validasi model dan ancaman, serta cara men-verifikasi kesuksesan dari aksi yang dilakukan?	☐	☐

BAB VI
IMPLEMENTASI

6.1. Penerapan Praktik Pengkodean yang Aman

Praktik pengkodean yang aman adalah serangkaian teknik dan metode yang digunakan untuk menghasilkan kode program yang tidak rentan terhadap serangan dan dapat meminimalkan risiko keamanan. Tujuannya adalah melindungi aplikasi dari berbagai jenis serangan, seperti injeksi SQL, *Cross-Site Scripting* (XSS), *Cross-Site Request Forgery* (CSRF), serta serangan lainnya. Banyak serangan terjadi karena kegagalan dalam memvalidasi masukan dari pengguna, aplikasi pihak ketiga, entitas eksternal, sistem basis data, atau proses lainnya. Oleh karena itu, seluruh masukan harus diperiksa dan divalidasi dengan ketat sebelum diproses untuk mencegah potensi ancaman. Beberapa standar dan pedoman untuk memastikan pengkodean yang aman antara lain:

- *OWASP Secure Coding Practices, Quick Reference Guide*
- *Oracle Secure Coding Guidelines for Java SE*
- *Software Engineering Institute (SEI) CERT Secure Coding Standards*.

6.2. Pengelolaan Risiko Komponen dan API *Third-Party*

Sebagian besar proyek pengembangan perangkat lunak menggunakan komponen pihak ketiga, baik yang bersifat *open-source* maupun *proprietary*. Penggunaan komponen pihak ketiga ini memberikan keuntungan dalam mengurangi waktu dan biaya pengembangan, serta meningkatkan kinerja aplikasi. Namun, komponen-komponen tersebut dapat memiliki kerentanan yang harus diidentifikasi dan dimitigasi secara tepat. Pengelolaan risiko keamanan terhadap komponen atau aset pihak ketiga dapat merujuk pada Keputusan Gubernur Jawa Barat Nomor: 048/Kep.438-Diskominfo/2023 tentang Manajemen Sistem Pemerintahan Berbasis Elektronik Pemerintah Daerah Provinsi Jawa Barat, khususnya pada Lampiran B, Bab III tentang Pengendalian Teknis Keamanan, Subbab 3.1 mengenai Manajemen Risiko.

Perancangan keamanan sistem harus diawali dengan kajian risiko keamanan sistem. Manajemen risiko dilakukan untuk menentukan tingkat potensi ancaman dan risiko yang terkait dengan perangkat lunak selama siklus pengembangan. Setelah risiko diidentifikasi, pengembang dapat menentukan kontrol yang tepat untuk mitigasi guna mengurangi atau menghilangkan risiko tersebut. Tabel 2.2 berikut memuat daftar periksa untuk kegiatan pengelolaan risiko.

6.3. Review Kode Program dan *Static Analysis Security Testing* (SAST)

Review kode program merupakan langkah penting untuk mendeteksi *bug* dalam implementasi. Proses review dapat dilakukan secara manual, yang memerlukan auditor dengan keahlian mendalam dalam mengidentifikasi kerentanan keamanan pada kode program. Jika perangkat daerah tidak memiliki sumber daya dengan keahlian tersebut, alat bantu seperti *Static Analysis Security Testing* (SAST) dapat digunakan untuk melakukan review keamanan kode secara otomatis. SAST bertujuan untuk mendeteksi pola kode yang tidak aman dan memastikan bahwa kebijakan pengkodean aman telah dipatuhi. Beberapa alat SAST bahkan dapat mendeteksi *bug* dalam kode dan memberikan rekomendasi atau alternatif yang lebih aman sebagai langkah mitigasi. Penggunaan SAST membantu memastikan bahwa kode program yang dihasilkan memiliki tingkat keamanan yang lebih tinggi sebelum diproduksi atau dipasang.

Daftar periksa untuk tahap implementasi disusun untuk mencakup kegiatan praktik pengkodean yang aman, pengelolaan risiko penggunaan komponen dan API *third-party*, serta reviu kode program dan *Static Analysis Security Testing* (SAST), sebagaimana ditunjukkan oleh Tabel 6.1.

Tabel 6.1. Daftar Periksa untuk Tahap Implementasi

No.	Item	Pertanyaan	Applied	Done
A. Praktik Pengkodean yang Aman				
A.1 Lingkungan Pengkodean				
1	*Build otomation*	Apakah berbagai pekerjaan dalam proses *build* telah diotomatisasi dan dipastikan bahwa kendali/*setting* keamanan tidak diabaikan? (PABW 2.1 Otomatisasi *Deployment* Aplikasi)	☐	☐
2	Format standar kode program	Apakah kode program memiliki *style* dan format standar (lebih baik jika menggunakan *tools* untuk pen-format-an)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
3	Aktivasi fitur keamanan	Apakah fitur keamanan yang tersedia sudah diaktifkan? (PABM Bab III Development)	☐	☐
4	*Compiler highest warning level*	Apakah kode sumber dikompilasi menggunakan tingkat peringatan tertinggi (*highest warning level*) dari compiler yang dipakai?	☐	☐
5	*Compiler setting*	Apakah pengaturan peringatan *compiler* diaktifkan untuk mendeteksi kode program yang tidak aman?	☐	☐
6	'*Clean build' concep*t	Apakah telah diimplementasikan konsep '*clean build*', di mana semua peringatan *compiler* dianggap *error* dan diperbaiki (kode program dimodifikasi), kecuali untuk peringatan yang *false positive* atau tidak relevan?	☐	☐
A.2. Penerapan Prinsip Keamanan dalam Kode				
1	Penerapan prinsip-prinsip keamanan dalam kode	Apakah kode sudah menerapkan berbagai prinsip keamanan yang dibahas pada Bab I? Contohnya: • Prinsip '*base access decisions on permission, not exclusion'* atau '*default deny*' (akses didasarkan atas apa yang boleh, alih-alih atas apa yang dilarang); • Prinsip '*least privilege*' (setiap proses dieksekusi dengan hak minimal);	☐	☐

No.	Item	Pertanyaan	Applied	Done
		• Prinsip '*segreation of duties*'.		
2	Enkripsi *cookie*	Apakah *cookie* yang sensitif di-enkripsi?	☐	☐
3	Autentikasi yang aman	Apakah kode program telah memperhitungkan autentikasi yang aman (*password* yang kuat, prosedur lupa *password*, dan sebagainya)?	☐	☐
4	Menghindari *unsafe functions*	Apakah pemrograman menghindari fungsi-fungsi yang tidak aman / *unsafe functions* (contoh: fungsi '*gets*' pada bahasa pemrograman C)?	☐	☐
5	*Restrict system-level resources*	Apakah akses pengguna terhadap *system-level resources* dibatasi?	☐	☐
6	Menghindari JavaScript dalam WebView	Apakah sudah dihindari penggunaan JavaScript dalam WebView? (PABM 3.13 Kerawanan WebView)	☐	☐
7	Pengelolaan memori aman	Apakah memori dikelola secara aman? (PABM 3.16 Penanganan Memori)	☐	☐
8	Efisiensi kode sumber	Apakah kode sumber sudah bersih dari pengulangan kode yang tidak diperlukan? (PABM 2.1.6 Penerapan *Software Design Principles*)	☐	☐
9	Penerapan prinsip desain	Apakah kode program sudah menerapkan prinsip-prinsip	☐	☐

No.	Item	Pertanyaan	Applied	Done
	berorientasi obyek	design berorientasi obyek (seperti SRP, OCP, LSP, ISP, dan DIP)? (PABM 2.1.6 Penerapan *Software Design Principles*)		
10	Penyederhanaan arsitektur dan logika aplikasi	Apakah kode program sudah menerapkan penyederhanaan arsitektur dan logika aplikasi (misalnya menggunakan *Jetpack Compose*)? (PABM 2.1.6 Penerapan *Software Design Principles*)	☐	☐
11	Penerapan struktur desain MVC	Apakah kode program menerapkan strukur desain MVC (*Model-View-Controller*)? (PABM 2.1.6 Penerapan *Software Design Principles*)	☐	☐
12	Hak minimal pada akun DB.	Apakah akun basis data diberikan hak minimal?	☐	☐
13	Fungsi yang memanggil *shell*	Apakah sudah dihindari menggunakan fungsi yang dapat memanggil perintah *shell*?	☐	☐
14	Pemeriksaan perintah dan parameter *shell*	Saat menggunakan fungsi yang dapat memanggil perintah *shell*, apakah sudah diperiksa semua variabel yang membentuk parameter *shell* dan dipastikan untuk mengeksekusi hanya perintah yang diizinkan?	☐	☐
A.3. Validasi				

No.	Item	Pertanyaan	Applied		Done
1	Validasi positif dan sanitasi/ *filter input* /masukan	Apakah semua masukan sudah divalidasi positif dan disanitasi/di-*filter*, baik dari pengguna, aplikasi *third-party*, infrastruktur, entitas/sumber eksternal, sistem database, atau proses lainnya? (PABW 3.5 Validasi Data dan Masukan/*Input*; PABM 3.14 Kerawanan Basis Data)	☐	☐	☐
2	*Server-side validation*	Apakah validasi di sisi klien (yang mudah untuk diakali) sudah dihindari dan diterapkan validasi pada sisi server? (PABW 3.5 Validasi Data dan Masukan/*Input*)	☐		☐
3	Penolakan *Invalid input*	Apakah semua masukan yang tidak valid selalu ditolak? (PABW 3.5 Validasi Data dan Masukan/*Input* dan 3.1 Penanganan Kesalahan dan Eksepsi)	☐		☐
4	Validasi HTTP *header*	Apakah dilakukan validasi terhadap HTTP *header*?	☐		☐
5	HTTP 1.0 nonaktif dan memakai HTTPS	Apakah protokol HTTP 1.0 sudah dinonaktifkan dan sudah menggunakan HTTPS? (PABW 2.3.1 Menonaktifkan Protokol HTTP 1.0, PABW 2.3.2 Menggunakan HTTPS)	☐		☐

No.	Item	Pertanyaan	Applied	Done
6	Pelacakan HTTP *request* nonaktif	Apakah metode pelacakan pada web server telah dinonaktifkan? (PABW 2.3.8 Menonaktifkan Pelacakan HTTP *Request*)	☐	☐
7	Tidak menulis HTTP *header* secara langsung	Apakah sudah dipastikan bahwa HTTP *header* tidak ditulis secara langsung, tetapi melalui API HTTP *header* yang disediakan oleh lingkungan eksekusi atau bahasa pemrograman?	☐	☐
8	Netraliasi *buffering* secara manual	Jika API HTTP *header* yang menawarkan netralisasi umpan (*buffering*) tidak tersedia, apakah *buffering* diterapkan secara manual?	☐	☐
9	Menghapus karakter baris	Apakah input teks eksternal menghapus semua karakter umpan baris (yaitu *new line* dan *carriage return*)?	☐	☐
10	*Buffer overflow validation*	Apakah dilakukan validasi data dalam hal tipe, panjang, format, dan *range* untuk menghindari *buffer overflow attacks*?	☐	☐
11	Sterilisasi data ke subsistem lain	Apakah setiap data yang dikirimkan ke subsistem lainnya seperti *shell* perintah, *database*, komponen COTS (*commercial off-the-shelf*) sudah dijamin kebersihannya terlebih dahulu?	☐	☐
12	Keamanan *runtime environment*	Apakah sudah dipastikan *runtime environment* aplikasi tidak rentan terhadap serangan validasi input	☐	☐

No.	Item	Pertanyaan	Applied	Done
		(PABW 3.14.2 *Cross-Site Scripting*/XSS, 3.14.4 *SQL Injection*, 3.14.5 *Command Injection*)		
13	Fitur kode dinamis	Apakah fitur kode dinamis, seperti *reguler expression*, digunakan dalam validasi input?	☐	☐
14	Pelindungan terhadap XSS	Apakah sudah dilakukan pelindungan terhadap akses yang mengandung konten skrip atau serangan XSS (*Cross-Site Scripting*)? (PABW 2.3.9 Menyetel *Cookie* dengan HttpOnly dan *Secure Flag* untuk Menangkal Serangan XSS, 2.3.11 Melakukan Perlindungan X-XSS, 3.14.2 *Cross-Site Scripting* (XSS))	☐	☐
15	Pelindungan terhadap *SQL Injection*	Apakah sudah dilakukan pelindungan dari serangan injeksi basis data? (PABW 3.14.4 *SQL Injection*)	☐	☐
16	*Placeholder* untuk SQL	Apakah semua pernyataan SQL menggunakan *placeholder?*	☐	☐
17	Penggunaan fungsi *escaping* dan lainnya dari mesin *database*	Apakah saat membuat pernyataan SQL melalui *concatenation,* digunakan fungsi khusus yang ditawarkan oleh mesin *database* untuk melakukan *escaping* dan memperbaiki literal pernyataan SQL dengan benar?	☐	☐

No.	Item	Pertanyaan	Applied	Done
18	Parameter tidak mengandung SQL	Apakah sudah dipastikan bahwa pernyataan/perintah SQL tidak ditulis langsung pada parameter yang akan dilewatkan ke aplikasi?	☐	☐
A.4. *Mail Header Injection*				
1	Nilai tetap pada *email header* dan *external input*	Apakah sudah digunakan nilai tetap untuk elemen *header* dan menempatkan semua *external input* ke badan email untuk mencegah serangan *phising* dengan cara memanipulasi *header*?	☐	☐
2	Penggunaan API untuk kirim *email*	Jika nilai tetap untuk elemen *header* tidak dapat diterapkan, apakah digunakan API pengiriman *email* yang ditawarkan oleh lingkungan atau bahasa pemrograman web?	☐	☐
3	*Email* tidak dicantumkan dalam HTML	Apakah sudah dipastikan untuk tidak mencantumkan alamat *email* secara langsung dalam HTML untuk mencegah serangan dari *email harvesting bots*?	☐	☐
4	Hapus karakter baris baru	Apakah semua karakter baris baru dari *input* teks eksternal sudah dihapus untuk menghindari *mail header injection*?	☐	☐
A.5. Pencegahan Serangan				
1	Pencegahan *hidden field manipulation*	Apakah sistem telah mencegah serangan manipulasi *field* tersembunyi (*hidden field manipulation*)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
2	Pencegahan *cookie poisoning*	Apakah sistem telah mencegah serangan *cookie poisoning*?	☐	☐
3	Pencegahan *parameter tampering*	Apakah sistem telah mencegah serangan pengubahan parameter (*parameter tampering*)?	☐	☐
4	Pencegahan *forced browsing*	Apakah sistem telah mencegah serangan jelajah paksa (*forced browsing*)?	☐	☐
5	Pencegahan *race condition*	Apakah sistem sudah dijaga terhadap kejadian *race condition* (*infinite loops, deadlocks, resource collisions*)?	☐	☐
6	Pencegahan CSRF	Apakah kode program telah memperhitungkan serangan *Cross Site Request Forgery* (CSRF)? (PABW 3.14.3 *Cross-Site Request Forgery* (CSRF))	☐	☐
7	Pemeriksaan kesesuaian pengarah vs. URL	Apakah dilakukan pemeriksaan terhadap kesesuaian antara pengarah (*header* HTTP yang mengidentifikasi sumber permintaan) dengan URL yang diharapkan, dan proses berlanjut hanya ketika URL benar?	☐	☐
8	Pemberitahuan otomatis untuk operasi penting	Ketika operasi penting tertentu telah dilakukan, apakah sistem memberitahu secara otomatis melalui alamat *email* yang telah ditentukan sebelumnya?	☐	☐

No.	Item	Pertanyaan	Applied	Done
9	*Escaping output*	Apakah sudah dilakukan *escaping* untuk semua *output* yang dikeluarkan ke halaman web? (PABW 3.8 *Output Encoding*)	☐	☐
10	Pengecekan alamat URL	Saat menuliskan URL dalam HTML, apakah hanya mengizinkan pola tertentu seperti http:// dan https://?	☐	☐
11	Tidak membuat tag <script> dinamis	Apakah sudah dipastikan untuk tidak membuat konten tag <script> ... </script> secara dinamis?	☐	☐
12	Tidak meng-impor *stylesheet*	Apakah sudah dipastikan bahwa sistem tidak mengizinkan mengimpor *stylesheet* dari situs web yang tidak jelas?	☐	☐
13	*Parsing* dan ekstrak input teks HTML	Jika ada input teks HTML, apakah sudah dibuat pohon *parse* dan diekstrak sehingga hanya elemen yang tidak mengandung skrip yang diizinkan?	☐	☐
14	Menghapus skrip dalam input teks HTML	Jika ada input teks HTML, apakah string skrip yang terkandung di dalamnya sudah dihapus?	☐	☐
15	Menentukan nilai *charset*	Apakah nilai parameter *charset* pada *header* jenis konten HTTP sudah diset dengan benar?	☐	☐
16	Setting atribut *cookie* dan metode TRACE	Apakah atribut *cookie* sudah diset “HttpOnly” dan setting metode TRACE sudah dinonaktifkan?	☐	☐

No.	Item	Pertanyaan	Applied	Done
		(PABW 2.3.8. Menonaktifkan Pelacakan HTTP *Request*, 2.3.9. Menyetel *Cookie* dengan HttpOnly dan *Secure Flag* untuk Menangkal Serangan XSS)		
A.6. Penanganan Kesalahan dan Pencatatan Log				
1	*Error detection and handling*	Apakah digunakan metode/mekanisme deteksi dan penanganan *error* atau kesalahan untuk mencegah kesalahan terprediksi dan tidak terduga serta menangani seluruh pengecualian yang tidak ditangani? (PABW 3.1 Penanganan *Error* dan Eksepsi, PABM 3.17 Penanganan Eror dan Eksepsi)	☐ ☐	☐
2	*Generic error handler* (GEH)	Apakah ada *generic error handler* atau *exception handler* untuk menangani kesalahan yang tidak diantisipasi sebelumnya?	☐	☐
3	Penghentian proses jika *error*	Apakah jika terjadi kesalahan fatal, aplikasi akan menghentikan proses agar kondisi tetap aman?	☐	☐
4	Pengaturan konten pesan kesalahan	Apakah konten pesan yang ditampilkan ketika terjadi kesalahan sudah diatur dengan baik (misalnya tidak menampilkan detail teknis atau internal dari kesalahan, serta memberikan petunjuk solusi)? (PABW 3.1 Penanganan Error dan Eksepsi, 3.2 *Logging*)	☐	☐

No.	Item	Pertanyaan	Applied	Done
5	Informasi yang dikecualikan dalam log	Apakah sudah dipastikan bahwa log tidak mencantumkan informasi yang dikecualikan? (PABW 3.14.11 *Security Logging and Monitoring Failures*)	☐	☐
6	Cakupan catatan dalam log	Apakah sudah diatur mengenai apa yang dicatat dalam log untuk mendukung upaya penyelidikan ketika terjadi insiden? (PABW 3.2 Logging)	☐	☐
7	Pelindungan terhadap log aplikasi	Apakah sudah diatur pelindungan log aplikasi dari akses dan modifikasi yang tidak sah? (PABW 3.14.11 *Security Logging and Monitoring Failures*)	☐	☐
8	Enkripsi log	Apakah data log yang disimpan dienkripsi untuk mencegah injeksi log? (PABW 3.14.11 *Security Logging and Monitoring Failures*)	☐	☐
9	Sinkronisasi waktu log	Apakah sudah dilakukan sinkronisasi sumber waktu sesuai dengan zona waktu dan waktu yang benar? (PABW 3.14.11 *Security Logging and Monitoring Failures*)	☐	☐
10	*Log* dan penelusuran	Apakah aplikasi menuliskan *log* pemrosesan agar dapat dilakukan penelusuran (*tracing*)?	☐	☐
A.7. Pengendalian Kode Berbahaya				

No.	Item	Pertanyaan	Applied	Done
1	Reviu/analisis otomatis dan manual	Apakah kode telah direviu/dianalisis baik secara manual maupun otomatis menggunakan *tool* analisis untuk melindungi aplikasi terhadap pola serangan dan penyalahgunaan? (PABW 4.1 *Static Application Security Testing* /SAST)	☐	☐
2	Kode sumber tidak mengandung kode berbahaya	Apakah sudah dipastikan kode sumber aplikasi dan pustaka tidak mengandung kode berbahaya dan fungsionalitas lain yang tidak diinginkan? (PABW 4.1 *Static Application Security Testing* /SAST)	☐	☐
3	Izin terkait fitur/ sensor mengenai privasi	Apakah sudah diatur izin terkait fitur atau sensor terkait privasi? (Umumnya sudah langsung diatur browser, seperti izin lokasi, izin kamera, serta izin *cookies*)	☐	☐
4	Pelindungan integritas	Apakah sudah diatur pelindungan integritas? (PABW 3.14.10 *Software and Data Integrity Failures*)	☐	☐
5	Mekanisme fitur pembaruan	Apakah mekanisme fitur pembaruan sudah diatur? (PABW 3.14.8 *Vulnerable and Outdated Components*)	☐	☐
A.8. Aspek Implementasi Keamanan Lainnya				
1	Reviu terhadap	Apakah *coding*/implementasi sudah direviu untuk memastikan	☐	☐

No.	Item	Pertanyaan	Applied	Done
	kebutuhan dan desain	bahwa kebutuhan dan desain keamanan yang telah dispesifikasikan pada tahap sebelumnya (termasuk berbagai fitur seperti enkripsi) sudah diakomodasi/ diterapkan?		
B. Pengelolaan Risiko Penggunaan Komponen dan API *Third-Party*				
1	Inventarisasi komponen	Apakah sudah dilakukan inventarisasi penggunaan komponen dan API *third-party*, baik yang *open-source* maupun *proprietary*?	☐	☐
2	Evaluasi dan seleksi komponen	Sebelum memutuskan penggunaan komponen atau API *third-party*, apakah sudah diuji untuk keamanan dan kualitas serta dipastikan bahwa vendor atau penyedia layanan memiliki reputasi yang baik?	☐	☐
3	Pemenuhan kebutuhan keamanan komponen	Apakah kebutuhan keamanan (termasuk mekanisme verifikasi integritas) untuk komponen, API, atau perangkat lunak yang dikerjasamakan dengan pihak ketiga sudah didefinisikan, dikomunikasikan, dan diuji?	☐	☐
4	Identifikasi kerentanan komponen	Apakah sudah dilakukan identifikasi kelemahan, kerentanan, atau isu keamanan komponen dan API *third-party* dengan melakukan scan kerentanan atau mengikuti	☐	☐

No.	Item	Pertanyaan	Applied	Done
		informasi terkini secara teratur, serta mendokumentasikannya?		
5	Mitigasi komponen	Apakah sudah dilakukan perencanaan dan pelaksanaan mitigasi terhadap kerentanan komponen dan API yang ditemukan (misalnya melalui *patch*)?	☐	☐
C. Reviu Kode Program dan *Static Analysis Security Testing* (SAST)				
1	Analisis tambahan	Apakah dilakukan analisis tambahan selain analisis otomatis dengan *tools* (tidak wajib tapi dianjurkan)?	☐	☐
2	Prioritas reviu	Apakah reviu difokuskan kepada komponen yang kritikal (ada prioritas dalam reviu)?	☐	☐
3	*Peer review*	Apakah kode program sudah diperiksa secara *peer review*?	☐	☐
4	*Root cause* dari temuan	Apakah temuan-temuan dalam reviu telah diidentifikasi akar penyebabnya (*root cause*) dan didokumentasikan?	☐	☐
5	Eksekusi saran *peer review*	Apakah saran-saran yang diberikan dalam *peer review* sudah dilaksanakan?	☐	☐
6	Penggunaan SAST *tools*	Apakah digunakan *Static Analysis Security Testing* (SAST) *tools* dalam pemeriksaan kode program?	☐	☐
7	Eksekusi saran SAST *tools*	Apakah saran-saran yang diberikan oleh SAST *tools* sudah dilaksanakan?	☐	☐

No.	Item	Pertanyaan	Applied	Done
8	Pendeteksian potensi cacat	Apakah alat bantu dapat secara otomatis menemukan potensi cacat dalam aplikasi?	☐	☐
9	Resolusi potensi cacat	Apakah resolusi terhadap potensi cacat sudah dilakukan?	☐	☐
10	Penanganan *false positives*	Apakah tim menemukan isu sebenarnya jika ada banyak *false positives* untuk kondisi tertentu?	☐	☐

BAB VII
PENGUJIAN DAN DEPLOYMENT

Tahap pengujian dan *deployment* bertujuan untuk memastikan bahwa perangkat lunak tidak hanya berfungsi sesuai spesifikasi, tetapi juga aman dari berbagai ancaman dan kerentanan. Jika metodologi *DevSecOps* diterapkan, pengujian dapat dilakukan langsung melalui berbagai alat yang dimiliki oleh tim, yang memungkinkan pendeteksian risiko serta penyelesaiannya dengan lebih cepat. Sebaliknya, jika menggunakan metode tradisional, pengujian akan mengikuti tahapan pengembangan biasa, disertai dengan pengujian keamanan aplikasi.

7.1. Pengujian

7.1.1. *Vulnerability Assessment* (VA) dan *Penetration Testing*

Vulnerability Assessment (VA) merupakan proses untuk mengidentifikasi dan mengevaluasi kelemahan atau celah dalam aplikasi, dengan tujuan menentukan tingkat risiko yang mungkin dihadapi serta memberikan rekomendasi untuk menguranginya.

Berbeda dengan VA, *penetration testing* (pentest) berfokus pada eksploitasi kelemahan yang telah diidentifikasi. Pentest adalah *black box testing* yang dilakukan pada sistem untuk mensimulasikan serangan seperti yang dilakukan oleh penyerang nyata. Pentest dapat dilakukan oleh tim internal atau pihak eksternal, dengan tujuan menemukan berbagai kerentanan, mulai dari *bugs* kecil hingga cacat desain yang memengaruhi pengkodean, konfigurasi sistem, dan kelemahan operasional lainnya.

7.1.2. *Dynamic Analysis Security Testing* (DAST)

Dynamic Analysis Security Testing (DAST) melakukan verifikasi terhadap perangkat lunak saat dijalankan (*run-time*). Pengujian ini memeriksa fungsionalitas dan kondisi aplikasi ketika komponen sistem diintegrasikan. DAST sering dilengkapi dengan paket serangan tertentu untuk mendeteksi masalah seperti *corrupted memory*, masalah hak akses, *injection*, dan kerentanan keamanan lainnya. DAST dapat dianggap sebagai bentuk otomatis dari *penetration testing* yang menggunakan alat-alat tertentu.

7.1.3. *Interactive Application Security Testing* (IAST)

Interactive Application Security Testing (IAST) adalah metode pengujian keamanan aplikasi yang dilakukan saat aplikasi dijalankan melalui pengujian otomatis, pengujian oleh manusia, atau aktivitas lainnya yang berinteraksi dengan fungsionalitas aplikasi. IAST menggabungkan pendekatan dari *Static Application Security Testing* (SAST) dan DAST, dengan lingkungan *DevOps* sebagai dasar otomasi pengujian.

7.1.4. Pengujian Keamanan Berbasis Risiko

Pengujian keamanan berbasis risiko mencakup dua strategi: pengujian fungsionalitas keamanan menggunakan teknik pengujian standar, dan pengujian yang didasarkan pada pola serangan serta hasil analisis risiko arsitektur. Pengujian berbasis risiko dapat dimulai sebelum perangkat lunak selesai dibangun, dan melibatkan pengujian unit *white box* serta *stubs*. Sementara itu, *penetration testing* dilakukan ketika perangkat lunak telah selesai dibangun, dengan metode *black box* untuk menguji dari luar ke dalam. Meskipun berbeda dalam pendekatan, kedua metode tetap berpedoman pada analisis risiko dan kebutuhan keamanan fungsional.

7.1.5. Manajemen Pengujian Keamanan

Manajemen pengujian keamanan mencakup perencanaan yang matang, pelaksanaan sistematis, dan evaluasi mendalam untuk memastikan perangkat lunak aman dari ancaman. Menggunakan langkah-langkah terstruktur serta alat dan teknik yang tepat memungkinkan organisasi mengurangi risiko keamanan dan menjaga kualitas perangkat lunak. Praktik terbaik seperti pengujian otomatis, pengelolaan kerentanan, serta pemantauan berkelanjutan adalah elemen kunci untuk mencapai tujuan ini.

7.2. *Deployment*

Pada tahap *deployment*, perangkat lunak yang telah dikembangkan dan diuji dipindahkan ke lingkungan produksi yang digunakan oleh pengguna akhir. Lingkungan produksi biasanya dikonfigurasi lebih ketat dibandingkan dengan lingkungan pengembangan atau pengujian. Pembatasan dalam lingkungan produksi dapat mencakup pembatasan port dan protokol, segmentasi jaringan, layanan yang dinonaktifkan, serta kontrol komponen lainnya.

Daftar periksa untuk tahap pengujian dan deployment disusun untuk mencakup kegiatan pengujian *vulnerability assessment* (VA) dan *penetration testing*, *Dynamic Analysis Security Testing* (DAST), *Interactive Application Security Testing* (IAST), pengujian keamanan berbasis risiko, manajemen pengujian keamanan, serta kegiatan deployment, sebagaimana ditunjukkan oleh Tabel 7.1 berikut.

Tabel 7.1. Daftar Periksa untuk Tahap Pengujian dan *Deployment*

No.	Item	Pertanyaan	Applied	Done
A. *Vulnerability Assessment* dan *Penetration Testing*				
1	*Vulnerability Assessment* (VA)	Apakah Vulnerability *Assessment* (VA) telah dilakukan untuk menguji level keamanan aplikasi?	☐	☐
2	Lingkup VA	Untuk aplikasi berbasis web, apakah lingkup VA termasuk kerentanan umum dalam aplikasi web?	☐	☐
3	Tindak lanjut VA	Apakah berbagai temuan dalam VA sudah ditindaklanjuti / diperbaiki?	☐	☐
4	*Penetration testing (pentest)*	Apakah *penetration testing* sudah dilakukan sebagai validasi final dari tindak lanjut berbagai isu keamanan?	☐	☐
5	Tindak lanjut *pentest*	Apakah berbagai temuan dalam *pentest* sudah ditindaklanjuti?	☐	☐
6	*Final application*	Apakah pengujian dilakukan pada lingkungan produksi dan konfigurasi final?	☐	☐
7	Lingkup pengujian	Apakah lingkup pengujian memasukkan Top 10 risiko keamanan (misalnya: *The OWASP*	☐	☐

No.	Item	Pertanyaan	Applied	Done
		Top 10 Most Critical Web Application Security Risks)?		
8	Penyediaan *provenance data*	Jika ada *provenance data* (metadata yang mengkonfirmasi keautentikan, sumber, serta riwayat data), apakah data tersebut tersedia bagi tim VA dan *pentest* untuk membantu mereka dalam memitigasi kerentanan?	☐	☐
B. Dynamic Analysis Security Testing (DAST)				
1	Penggunaan DAST	Apakah DAST *tools* digunakan untuk memeriksa keamanan saat program berjalan? (PABW 4.2 Dynamic Application Security Testing (DAST))	☐	☐
2	*Scan running software*	Apakah proses *scan* terhadap perangkat lunak yang sedang berjalan telah dilakukan untuk menemukan kelemahan? (PABW 4.2 Dynamic Application Security Testing (DAST))	☐	☐
3	Resolusi potensi cacat	Apakah resolusi terhadap kelemahan atau potensi cacat sudah dilakukan? (PABW 4.2 Dynamic Application Security Testing (DAST))	☐	☐
C. *Interactive Application Security Testing* (IAST)				
1	Pengujian dengan IAST	Apakah dilakukan *Interactive Application Security Testing*? (PABW 4.3 *Interactive Application Security Testing* (IAST))	☐	☐

No.	Item	Pertanyaan	Applied	Done
D. Pengujian Keamanan Berbasis Risiko				
1	Pengujian berdasarkan risiko	Apakah pendekatan dan pemilihan kasus pengujian perangkat lunak didasarkan atas analisis risiko?	☐	☐
2	Pola serangan dan kasus penyalahgunaan	Apakah pengujian risiko sudah memasukkan pola serangan dan kasus penyalahgunaan?	☐	☐
3	Pertimbangan kerentanan sebelumnya	Apakah pengujian sudah mempertimbangkan berbagai kerentanan dan insiden yang dilaporkan sebelumnya?	☐	☐
E. Manajemen Pengujian Keamanan				
1	Rencana pengujian	Apakah pengujian telah direncanakan dan dokumen rencana pengujian telah dibuat?	☐	☐
2	Pengujian fungsi keamanan	Apakah pengujian fungsionalitas keamanan dengan teknik-teknik pengujian standar telah dilakukan?	☐	☐
3	*Abuse case*	Apakah pengujian dilakukan untuk kasus penyalahgunaan penggunaan fitur?	☐	☐
4	*Root cause analysis*	Apakah telah dilakukan analisis terhadap akar penyebab kerentanan (*root cause analysis*)?	☐	☐
5	Otomatisasi	Apakah pengujian sedapat mungkin diotomatisasi (misalnya menggunakan *fuzz testing tools* untuk menguji penanganan input)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
6	*White box testing*	Apakah *white box testing* (+*unit testing*) dilakukan untuk menganalisa data, kendali, aliran informasi, *exception, error hadling*, serta *use case* dan *misuse case*?	☐	☐
7	*Black box testing*	Apakah *black box testing* dilakukan untuk memeriksa perilaku eksternal aplikasi, pemenuhan kebutuhan, protokol, antarmuka, termasuk *vulnerability scanning*?	☐	☐
8	Tes lainnya: *developer testing*	Apakah dilakukan tes lainnya, seperti *integration test, regression test,* dan *software test*?	☐	☐
9	Tidak menggunakan data sensitif	Apakah data yang diuji bukan merupakan data sensitif? (pengujian data sensitif pada lingkungan pengujian harus mendapatkan approval dari *senior management*)	☐	☐
10	Kepatuhan terhadap kebijakan	Jika ada kebijakan/SOP pengujian (misalnya harus dalam lingkungan *sandbox*), apakah pengujian telah mengikuti kebijakan/SOP tersebut?	☐	☐
11	Pendokumentasian pengujian	Apakah telah dilakukan pendokumentasian/pelaporan terhadap pengujian, termasuk *lesson learned* dari hasil pengujian?	☐	☐
12	Kerentanan yang diterima	Apakah sudah ditentukan tingkat kerentanan yang dapat diterima?	☐	☐
13	Validasi model ancaman	Apakah model ancaman divalidasi, dengan kata lain pengujian didasarkan atas daftar ancaman	☐	☐

No.	Item	Pertanyaan	Applied	Done
		yang sama yang digunakan ketika melakukan pemodelan ancaman?		
14	Area pengujian keamanan	Apakah area pengujian keamanan sudah mencakup (1) validasi input, (2) kendali cacat injeksi, (3) kendali serangan *sripting*, (4) kendali kenirsangkalan atau *non-repudiation*, (5) kendali *spoofing*, (6) kendali penangan kesalahan dan *exception*, (7) kendali eskalasi *privileges*?	☐	☐
15	*Dummy data*	Apakah data pengujian menggunakan *dummy data* (bukan data produksi) yang tidak memiliki nilai bisnis?	☐	☐
16	Uji penerimaan (*acceptance test*)	Apakah dilakukan uji penerimaan (*acceptance test*) untuk menunjukkan bahwa perangkat lunak siap untuk digunakan serta memenuhi semua kebutuhan fungsional dan keamanan?	☐	☐
F. Deployment				
1	Kriteria kelengkapan untuk reviu	Apakah sudah dibuat kriteria kelengkapan untuk memeriksa bahwa semua kebutuhan fungsional dan keamanan sudah terpenuhi (contoh: pemenuhan model ancaman, reviu kode untuk kerentanan, kelengkapan dokumentasi, dan penyelesaian pengujian keamanan)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
2	*Hardening* (penguatan)	Apakah sudah dilakukan *hardening* (penguatan) pada sistem operasi, aplikasi, dan komponen melalui konfigurasi, *update*, dan *patch*?	☐	☐
3	Cek integritas saat instalasi atau *startup*	Apakah dilakukan pengecekan integritas saat instalasi dan startup perangkat lunak, atau secara berkala?	☐	☐
4	Manajemen rilis	Apakah telah dilaksanakan manajemen rilis untuk memastikan pengujian, analisis risiko, dan persetujuan dari pihak-pihak yang terlibat?	☐	☐
5	Penyimpanan versi rilis	Apakah file, gambar, dan lain-lain dari versi rilis sudah disimpan dalam *repository* dan aksesnya diatur *read only*?	☐	☐
6	Persetujuan rilis	Apakah semua pihak yang memiliki otoritas sudah menyetujui rilis perangkat lunak?	☐	☐
7	Penerimaan risiko	Apakah semua pihak yang memiliki otoritas sudah mengetahui dan menerima risiko residual yang ada (risiko harus diterima oleh pemilik bisnis dan bukan oleh pejabat di bagian TI)?	☐	☐
8	Penandatanganan aplikasi	Apakah aplikasi telah ditandatangani dengan sertifikat yang valid?	☐	☐

No.	Item	Pertanyaan	Applied	Done
		(PABM 3.12 Kerawanan Proteksi Binary)		
9	Aplikasi dalam mode rilis	Apakah sudah dipastikan bahwa aplikasi dalam mode rilis? (PABM 3.11 Kerawanan Konfigurasi Keamanan)	☐	☐
10	Penghapusan simbol *debugging*	Apakah simbol *debugging* dari *native binary* sudah dihapus? (PABM 3.11 Kerawanan Konfigurasi Keamanan)	☐	☐
G. Manajemen Konfigurasi				
1	Pemeliharaan konfigurasi *baseline*	Apakah konfigurasi *baseline* dan inventori sistem informasi (termasuk *hardware, software, firmware*, dan dokumentasi) telah dibuat dan dipelihara sepanjang siklus hidup pengembangan sistem?	☐	☐
2	Penetapan dan pemberlakuan konfigurasi	Apakah *setting* konfigurasi keamanan untuk produk TI pada sistem sudah ditetapkan dan diberlakukan?	☐	☐
3	*SSH port*	Apakah *port* SSH (22) sudah dibatasi hanya untuk *host* tertentu atau jaringan internal organisasi?	☐	☐
4	*Web port*	Apakah seluruh akses ke aplikasi hanya dibuka untuk *port* 80 dan 443?	☐	☐
5	*Application port*	Apakah grup aplikasi dalam web hanya dibuka untuk *port* tertentu saja (misalnya 8080)?	☐	☐

No.	Item	Pertanyaan	Applied	Done
6	*Database port*	Apakah grup *database* hanya dibuka untuk *port* yang dibutuhkan saja (misalnya 3306)?	☐	☐
7	Konfigurasi layanan API dan *web service*	Untuk aplikasi berbasis web, apakah layanan web untuk API dan *web service* telah dikonfigurasi keamanannya? (PABW 3.11 API)	☐	☐
8	Konfigurasi sesuai rekomendasi	Apakah *server* sudah dikonfigurasi sesuai rekomendasi dari *server* aplikasi dan *framework* aplikasi yang digunakan (PABW 2.3 *Web Server Hardening*)	☐	☐
9	Dokumentasi dan *backup* konfigurasi	Apakah konfigurasi sudah didokumentasikan dan disalin beserta semua dependensinya? (PABW 3.14.8 *Vulnerable and Outdated Components*)	☐	☐
10	Penghapusan konfigurasi dan hal lainnya yang tidak perlu	Apakah fitur, dokumentasi, sampel, dan konfigurasi yang tidak diperlukan sudah dihapus? (PABW 2.3 *Web Server Hardening* dan 3.14.7 *Security Misconfiguration*)	☐	☐
11	Validasi integritas aset	Jika aset aplikasi diakses secara eksternal, apakah integritas aset divalidasi? (PABW 3.14.10 *Software and Data Integrity Failures*)	☐	☐

No.	Item	Pertanyaan	Applied	Done
12	Respon aplikasi dan konten yang aman	Apakah respons aplikasi dan konten yang digunakan sudah aman? (PABW 3.1 Penanganan *Error* dan Eksepsi)	☐	☐
13	Pelindungan akan *clickjacking*	Apakah X-Frame-Options pada *web server* sudah dikonfigurasi untuk mencegah *clickjacking*?	☐	☐

BAB VIII
OPERASI DAN PEMELIHARAAN

Pada tahap operasi dan pemeliharaan, tim pengembang keamanan secara berkala melakukan pemeriksaan keamanan terhadap perangkat lunak dan lingkungannya. Selain itu, tim melakukan pembaruan (*update*) untuk mengakomodasi kebutuhan yang terus berkembang, serta memastikan kompatibilitas dengan berbagai *tools* yang digunakan selama semua tahapan pengembangan. Kegiatan yang dilakukan dalam tahap ini umumnya mencakup:

1. Pendefinisian operasi dan pemeliharaan yang aman;
2. Pemantauan kendali keamanan;
3. Pengungkapan dan penanganan kerentanan; dan
4. Penyusunan prosedur untuk menanggapi insiden.

8.1. Pendefinisian Operasi dan Pemeliharaan yang Aman

Pendefinisian operasi dan pemeliharaan yang aman mencakup penerapan praktik serta proses yang dirancang untuk menjaga keamanan perangkat lunak selama fase operasional. Keamanan jaringan dapat diintegrasikan dengan keamanan perangkat lunak untuk meningkatkan efektivitas pertahanan. Mengidentifikasi perilaku penyerang dan memahami kerentanan perangkat lunak adalah kunci dalam mempertahankan keamanan. Pengetahuan yang diperoleh dari fase ini juga dapat digunakan sebagai masukan bagi tahapan lain dalam *Secure SDLC*..

8.2. Pemantauan Kendali Keamanan

Pemantauan kendali keamanan merupakan proses berkelanjutan yang bertujuan untuk memastikan kontrol keamanan pada perangkat lunak dan infrastruktur pendukung berfungsi secara efektif. Aktivitas yang dilakukan meliputi pengumpulan data dari *log* atau sensor, analisis data *log*, deteksi ancaman, dan pelaporan. Implementasi dapat dilakukan melalui *dashboard* yang menyajikan visualisasi data keamanan yang mudah dimengerti untuk memudahkan pengambilan keputusan.

8.3. Pengungkapan dan Penanganan Kerentanan

Walaupun perangkat lunak telah dikembangkan sesuai dengan pedoman *Secure SDLC*, tidak ada produk yang sepenuhnya aman karena lingkungan terus berubah. Oleh karena itu, unit kerja perlu mengembangkan proses yang efektif untuk mengungkap dan menangani kerentanan guna mendukung penyelesaian

masalah yang ditemukan. Selain itu, kemajuan penanganan kerentanan harus selalu diinformasikan kepada seluruh *stakeholders* terkait.

8.4. Penyusunan Prosedur Tanggapan terhadap Insiden

Terlepas dari penerapan *Secure SDLC*, perangkat daerah harus siap menghadapi serangan yang mungkin tidak dapat dihindari. Dibutuhkan pembentukan tim tanggap insiden seperti *Computer Security Incident Response Team* (CSIRT) serta penyusunan *Incident Response Plan* (IRP) sebagai langkah persiapan. Pengalaman dan pelajaran yang diperoleh dari insiden keamanan aktual harus digunakan untuk memperbaiki dan memperkuat siklus pengembangan perangkat lunak di masa mendatang.

Daftar periksa untuk tahap operasi dan pemeliharaan disusun untuk mencakup kegiatan pendefinisian operasi dan pemeliharaan yang aman, pemantauan kendali keamanan, pengungkapan dan penanganan kerentanan, serta penyusunan prosedur untuk menanggapi insiden. Tabel 8.1 berikut memperlihatkan daftar periksa untuk tahap operasi dan pemeliharaan.

Tabel 8.1. Daftar Periksa untuk Tahap Operasi dan Pemeliharaan

No.	Item	Pertanyaan	Applied	Done
A. Pendefinisian Operasi dan Pemeliharaan yang Aman				
1	Penerapan *control*/kendali untuk keamanan operasional	Apakah diterapkan *control*/kendali keamanan (proses, kebijakan, perangkat, praktik, mekanisme, *tools*, personil, atau aksi yang memodifikasi risiko) untuk menjaga agar ketahanan perangkat lunak berada pada level risiko yang dapat diterima?	☐	☐
2	Kecukupan SDM	Apakah alokasi sumber daya cukup untuk melindungi sistem informasi?	☐	☐
3	*System Availability*	Apakah sudah ditetapkan target tingkat ketersediaan / *availability* sistem (misalnya 99%)?	☐	☐
4	Redundansi	Untuk sistem dengan tingkat ketersediaan tinggi (*high*	☐	☐

No.	Item	Pertanyaan	Applied	Done
		availability), apakah sudah ada redundansi atau solusi *fault-tolerant*?		
5	*Establish and response to pre-defined threshold*	Apakah sudah ditetapkan batas maksimal/minimal ketersediaan sumber daya tertentu (mis. ruang *hardisk*, utilitas memori) dan rencana tindak lanjut jika sumber daya lebih/kurang dari *threshold*?	☐	☐
6	*Remote access managemen t*	Apakah telah diterapkan enkripsi, autentikasi yang kuat (seperti *multi-factor authentication*), serta perangkat dan jaringan yang aman untuk akses *remote*?	☐	☐
7	*Disposal*/ Penghapus an data	Apakah data rahasia sudah dihapus permanen (*irrecovably*) dari perangkat, ketika perangkat dibuang, dihancurkan, atau berpindah penggunaan/kepemilikan?	☐	☐
8	Penyertaan *hash* kriptografis	Apakah file-file yang dirilis menyertakan *hash* kriptografis atau *code signing* untuk pemeriksaan integritas?	☐	☐
9	Pelatihan bagi pengguna	Apakah pengguna mendapatkan pelatihan pemakaian aplikasi sebelum hak akses terhadap aplikasi tersebut diberikan?	☐	☐
10	Pelatihan pengguna ketika ada perubahan	Apakah pelatihan bagi pengguna kembali dilakukan ketika ada perubahan besar pada aplikasi?	☐	☐
B. Pemantauan Kendali Keamanan				

No.	Item	Pertanyaan	Applied	Done
1	Pemantauan kendali keamanan	Apakah kendali keamanan sistem dipantau secara terus menerus untuk menjamin efektivitasnya?	☐	☐
2	Log aplikasi	Apakah log aplikasi dipelihara dan di-*backup*?	☐	☐
3	*Backup*	Apakah sistem dan data sudah di-*backup* secara berkala?	☐	☐
4	Pemantauan log untuk *zero-day*	Apakah log sudah diperiksa untuk mendeteksi kemungkinan terjadinya *zero-day*?	☐	☐
5	*Intrusion detection* dan monitor aktivitas yang tidak biasa	Apakah sistem/administrator dapat mendeteksi/memonitor aktivitas yang tidak biasa atau perilaku abnormal (misalnya transaksi yang terjadi berulang kali dalam waktu singkat) melalui mekanisme tertentu (misalnya pemeriksaan *file* log)? (PABW 3.2 *Logging*)	☐	☐
6	Pemantauan sistem dan komunikasi	Apakah sistem dan komunikasi (yaitu informasi yang ditransmisikan atau diterima organisasi) sudah dipantau, dikendalikan, dan dilindungi?	☐	☐
7	*Performance monitoring*	Apakah kinerja program dipantau secara teratur?	☐	☐
8	*Patch*, *update*, dan lainnya	Apakah *patch*, *update* komponen dan pemeliharaan lainnya dilakukan baik secara periodik maupun sewaktu-waktu?	☐	☐
9	Pemantauan dan tindakan berdasarka	Apakah peringatan (*alerts*) keamanan sistem dan saran yang diberikan telah dipantau, serta	☐	☐

No.	Item	Pertanyaan	Applied	Done
	n alert sistem	diambil tindakan/respon yang tepat?		
10	Pengelolaan *data center*	Untuk perangkat daerah yang memiliki *data center* sendiri, apakah telah dilakukan pengelolaan *data center* dengan memperhatikan: – Kecukupan perangkat pendukung seperti redundansi *power*, koneksi internet, dan pendingin. – Kendali lingkungan seperti detektor asap/panas dan *sprinkler system*. – DRC (*Disaster Recovery Center*). – Pemantauan terhadap kendali lingkungan dan keamanan fisik?	☐	☐
11	Masa aktif komponen	Jika memakai komponen eksternal, apakah sudah dipastikan bahwa komponen tersebut masih berada dalam masa aktif dan belum mencapai *end of life*?	☐	☐
C. Pengungkapan dan Tanggapan terhadap Kerentanan				
1	Meng-identifikasi kerentanan	Apakah tim secara teratur melakukan identifikasi kerentanan/cacat sistem dalam keseluruhan siklus aplikasi?	☐	☐
2	Menilai risiko kerentanan	Jika teridentifikasi kerentanan, apakah dilakukan penilaian risiko terhadap kerentanan tersebut dalam hal kemungkinan eksploitasi, dampak, atau penilaian lainnya?	☐	☐
3	Melaporkan kerentanan	Apakah tim secara teratur membuat laporan mengenai berbagai kerentanan atau *bug* keamanan baru yang telah diidentifikasi?	☐	☐

No.	Item	Pertanyaan	Applied	Done
4	Mengatasi kerentanan	Apakah kerentanan sudah ditangani/dikoreksi (untuk kerentanan dengan risiko yang tidak dapat diterima), misalnya melalui *patch*?	☐	☐
5	Mitigasi sementara	Untuk kerentanan yang tidak dapat segera diatasi, apakah dilakukan mitigasi sementara sampai dengan solusi tersedia?	☐	☐
6	*Shared lessons learned*	Apakah identifikasi kerentanan, praktik penanganan, dan penggunaan alat bantu keamanan yang merupakan *best practices* dibagikan kepada tim-tim lain untuk pembelajaran bersama (misalnya dalam bentuk buku atau informasi *online*) dan selalu di-*update*?	☐	☐
7	*Update* informasi *terbaru*	Apakah tim mengikuti perkembangan informasi terbaru dari *tools*, teknik, dan ancaman (*threats*)?	☐	☐
8	*Update* informasi kerentanan	Apakah tim secara teratur memeriksa jika ada kerentanan terbaru yang dipublikasikan mengenai modul/komponen/layanan *software* tertentu dan belum diperbaiki oleh *vendor*-nya, serta memahami bagaimana kerentanan tersebut dapat dieksploitasi?	☐	☐
9	Update *progress* untuk	Apakah *stakeholders* selalu terinformasikan dalam hal	☐	☐

No.	Item	Pertanyaan	Applied	Done
	stakeholders	kerentanan dan progress penanganan?		
10	Pemberitahuan kerentanan ke pengguna	Apakah kerentanan dan bagaimana cara mengatasinya (misalnya melalui *patches*) diinformasikan kepada pengguna?	☐	☐
D. Pendefinisian Prosedur Tanggapan terhadap Insiden				
1	Agen CSIRT	Apakah sudah ada agen CSIRT di perangkat daerah berikut dengan uraian tugas dan tanggung jawabnya?	☐	☐
2	Penentuan insiden	Apakah anggota tim sudah dapat menentukan bahwa sebuah *event* (peristiwa) merupakan insiden dan membedakannya dari peristiwa biasa dan *alert* (peringatan)?	☐	☐
3	Rencana penanganan insiden	Apakah sudah ada dokumen perencanaan dalam bentuk SOP (*Standard Operation Procedure*) atau bentuk lainnya untuk menghadapi insiden?	☐	☐
4	Kelengkapan rencana penanganan insiden	Apakah dokumen perencanaan sudah mencakup *contact person* ketika terjadi kondisi darurat keamanan, prosedur mitigasi, prosedur tanggapan dan komunikasi kepada kastemer, prosedur perbaikan kerentanan, serta analisis *post-incident*?	☐	☐
5	Peran dan tanggung jawab	Apakah dokumen perencanaan penanganan insiden sudah mencakup peran dan tanggung jawab dari staf atau pihak eksternal?	☐	☐

No.	Item	Pertanyaan	Applied	Done
6	Pengujian *Incident Response Plan*	Apakah sudah dilaksanakan pengujian/simulasi terhadap IRP (*Incident Response Plan*)?	☐	☐
7	Kapabilitas penanganan insiden	Apakah sudah dibangun kapabilitas penanganan insiden yang mencakup aktivitas persiapan yang cukup, deteksi, analisis, resolusi, pemulihan, dan respon pengguna?	☐	☐
8	Laporan insiden	Apakah insiden yang terjadi sudah ditelusuri, didokumentasikan, dan dilaporkan kepada petugas atau pihak tertentu?	☐	☐
9	Rencana *contingency* atau *Disaster Recovery Plan*	Apakah DRP (*Disaster Recovery Plan*), yaitu rencana tanggap darurat, operasi *backup*, dan pemulihan bencana untuk menjamin ketersediaan sumberdaya informasi kritikal dan keberlangsungan operasi dalam situasi darurat sudah dibuat, dipelihara, dan diimplementasikan/diuji?	☐	☐

BAB IX
EVALUASI

9.1. Audit dan Penilaian Keamanan

Audit merupakan proses penting dalam penyusunan catatan historis mengenai aktivitas pengguna dalam sistem. Jejak audit ini berguna dalam mendeteksi perubahan yang dilakukan oleh pengguna tidak berwenang atau perubahan tidak sah yang dilakukan oleh pengguna yang berwenang, yang dapat dianggap sebagai pelanggaran terhadap integritas sistem. Selain itu, audit juga mencakup pemeriksaan terhadap kebijakan, proses kerja, manajemen risiko, dan kendali keamanan yang ada dalam organisasi.

9.2. Evaluasi Insiden dan Kerentanan

Evaluasi insiden dilakukan untuk memahami akar penyebab insiden, dampaknya terhadap sistem, serta tindakan pencegahan yang diperlukan agar insiden serupa tidak terjadi di masa depan. Evaluasi kerentanan bertujuan memastikan bahwa perangkat lunak yang dikembangkan telah terlindungi dari ancaman yang dapat dieksploitasi oleh pihak-pihak yang tidak bertanggung jawab, serta untuk mengidentifikasi dan mengatasi kelemahan yang ada.

9.3. Tindak Lanjut Hasil Evaluasi

Tindak lanjut dari hasil evaluasi meliputi serangkaian langkah yang diambil untuk memastikan bahwa tindakan korektif atau perbaikan telah diterapkan secara efektif. Langkah ini bertujuan untuk memperkuat keamanan perangkat lunak secara berkelanjutan. Selain itu, tindakan korektif memastikan bahwa organisasi dapat belajar dari insiden dan kerentanan yang terjadi, sehingga risiko ancaman di masa depan dapat diminimalkan dan dicegah secara lebih efektif.

Daftar periksa untuk tahap evaluasi disusun untuk mencakup kegiatan audit dan penilaian keamanan, evaluasi insiden dan kerentanan, serta tindak lanjut terhadap hasil evaluasi.

Tabel 9.1. Daftar Periksa untuk Tahap Evaluasi

No.	Item	Pertanyaan	Applied	Done
A. Audit dan Penilaian Keamanan				
1	Pelaksanaa naudit	Apakah audit dilakukan secara berkala untuk mendapatkan gambaran mengenai efektivitas kebijakan/aturan, proses kerja, pelaporan, manajemen risiko, dan kendali keamanan?	☐	☐
2	Manajemen rekaman audit	Apakah rekaman audit sistem telah dibuat, dilindungi, dan dijaga untuk kebutuhan pemantauan, analisis, investigasi, dan pelaporan aktivitas yang melanggar hukum, tidak sah, atau tidak normal?	☐	☐
3	Pelacakan aktivitas individu	Apakah aksi pengguna individu dapat secara unik dilacak sehingga mereka dapat dimintau pertanggungjawaban atas aksinya?	☐	☐
4	Penilaian kendali keamanan secara berkala	Apakah secara berkala, sudah dilakukan penilaian *control*/ kendali keamanan dalam sistem untuk melihat apakah kendali tersebut efektif?	☐	☐
5	Evaluasi otorisasi	Apakah sudah dilakukan evaluasi otorisasi pada sistem dan semua koneksi yang terhubung dengannya?	☐	☐
6	Reviu pengguna	Apakah dilaksanakan reviu secara berkala terhadap akun dan hak akses pengguna?	☐	☐
7	Audit informasi	Apakah audit data atau informasi dilaksanakan secara berkala?	☐	☐

No.	Item	Pertanyaan	Applied	Done
8	Evaluasi komponen *third-party*	Apakah audit memasukkan reviu dan evaluasi terhadap komponen pihak ketiga dalam hal pemenuhan terhadap kebutuhan/harapan?	☐	☐
B. Evaluasi Insiden dan Kerentanan				
1	*Lesson learned* dari laporan insiden	Apakah laporan insiden sudah memuat *lesson learned* untuk menjadi bahan pertimbangan siklus Secure SDLC berikutnya?	☐	☐
2	*Root cause analysis*	Untuk menjaga kerentanan tidak terjadi berulang pada produk baru atau produk hasil revisi, apakah sudah dilakukan analisis akar penyebab dan *lesson learned* dari kerentanan tersebut?	☐	☐
3	Memperhitungkan *root cause* dalam Secure SDLC	Apakah hasil analisis akar penyebab dan *leason learned* dari kerentanan sudah diperhitungkan dalam siklus pengembangan *software* berikutnya (misalnya menambahkan urutan langkah proses)?	☐	☐
C. Tindak Lanjut Hasil Evaluasi				
1	Tindak lanjut evaluasi insiden	Apakah telah dilakukan tindakan korektif/perbaikan berdasarkan *lesson learned* dari laporan/evaluasi insiden?	☐	☐
2	Perbaikan kerentanan	Apakah sudah dibangun dan diimplementasikan rencana aksi untuk memperbaiki kekurangan, serta mengurangi atau menghilangkan kerentanan?	☐	☐

No.	Item	Pertanyaan	Applied	Done
3	Tindak lanjut lainnya dari audit/evaluasi	Apakah saran-saran perbaikan dan tindakan korektif lainnya berdasarkan hasil audit atau evaluasi telah dilaksanakan?	☐	☐

BAB X
PENUTUP

Pedoman Teknis *Secure SDLC* merupakan panduan bagi Perangkat Daerah di lingkungan Pemerintah Daerah Provinsi Jawa Barat dalam mengintegrasikan praktik-praktik keamanan ke dalam siklus pengembangan perangkat lunak. Pedoman ini bertujuan untuk membantu Perangkat Daerah mendeskripsikan tugas-tugas yang harus dilaksanakan serta teknik-teknik pengamanan yang harus dipilih melalui daftar periksa pada setiap tahapan *Secure SDLC*, guna menjamin pengembangan perangkat lunak yang aman dan terlindungi dari berbagai ancaman.

Penerapan Pedoman Teknis *Secure SDLC* secara optimal memerlukan kesadaran dan kompetensi terkait keamanan informasi di kalangan personil Perangkat Daerah maupun tim pengembang. Kesadaran dan kompetensi tersebut dapat dibangun melalui edukasi dan pelatihan yang diselenggarakan oleh instansi atau perangkat daerah yang bertanggung jawab dalam bidang persandian dan keamanan informasi, maupun melalui pelatihan yang diselenggarakan secara mandiri oleh Perangkat Daerah terkait.

a.n. GUBERNUR JAWA BARAT
SEKRETARIS DAERAH,

Ditandatangani secara elektronik oleh:
SEKRETARIS DAERAH PROVINSI JAWA BARAT,

Drs. HERMAN SURYATMAN, M.Si.
Pembina Utama