MEMBANGUN APLIKASI YANG AMAN: SECURE SOFTWARE DEVELOPMENT LIFECYCLE (SSDL)

Oleh:
Doddy Ferdiansyah
R. Rizal Isnanto
Jatmiko E. Suseno

MEMBANGUN APLIKASI YANG AMAN: SECURE SOFTWARE DEVELOPMENT LIFECYCLE (SSDL)

Oleh:
Doddy Ferdiansyah
R. Rizal Isnanto
Jatmiko E. Suseno

Diterbitkan oleh
ASOSIASI DOKTOR Sistem Informasi INDONESIA
2025

Membangun Aplikasi yang Aman:
Secure Software Development Lifecycle (SSDL)

Oleh:
Doddy Ferdiansyah
R. Rizal Isnanto
Jatmiko E. Suseno

Uk. 15,5cm x 23cm (xii + 205hlm)

ISBN: 978-623-10-7157-6

Diterbitkan oleh

ASOSIASI DOKTOR
Sistem Informasi
INDONESIA

Editor: Dr. Usman Ependi, S.Kom., M. Kom.

Edisi Januari 2025

Hak Cipta dilindungi undang-undang
Dilarang memperbanyak sebagian atau seluruh isi buku ini dalam bentuk apa pun, baik secara elektronik maupun mekanik, termasuk memfotokopi, merekam atau dengan menggunakan sistem penyimpanan, tanpa izin tertulis dari Penulis.

KATA SAMBUTAN I

Ucapan syukur atas karunia dari Allah subhanahu wa ta'ala atas terbitnya buku berjudul "Membangun Aplikasi yang Aman: Secure Software Development Lifecycle (SSDL)" yang ditujukan untuk memberikan pemahaman yang mendalam tentang pentingnya keamanan dalam pengembangan perangkat lunak.

Teknologi informasi berkembang dengan pesat pada era digital saat ini berimplikasi keamanan aplikasi menjadi aspek yang sangat krusial. Setiap hari, kita berhadapan dengan berbagai ancaman yang dapat merusak integritas, kerahasiaan, dan ketersediaan data. Oleh karena itu, pemahaman tentang Secure Software Development Lifecycle (SSDL) sangatlah penting bagi para pengembang, pemangku kepentingan, serta semua pihak yang terlibat dalam proses pengembangan perangkat lunak.

Buku ini menyajikan konsep dan praktik baik dalam membangun aplikasi yang aman, dimulai dari pemahaman konsep perangkat lunak yang aman, perancangan perangkat lunak yang aman, sampai dengan praktik keamanan di tempat kerja. Sebagai pelengkap pada bagian akhir buku ini disajikan studi kasus dan diskusi dengan menampilkan beberapa skenario kasus terkini. Melalui pendekatan yang sistematis dan terstruktur, diharapkan pembaca dapat memahami tahapan-tahapan SSDL dan bagaimana cara mengimplementasikannya dalam proyek pengembangan yang mereka jalani.

Kami mengucapkan terima kasih kepada semua pihak yang telah berkontribusi dalam penyusunan buku ini. Semoga buku ini dapat menjadi sumber informasi yang bermanfaat dan dapat meningkatkan kesadaran serta pengetahuan kita tentang pentingnya membangun aplikasi yang aman. Selamat membaca dan semoga bermanfaat.

Prof. Dr. Ir. R. Rizal Isnanto, S.T., M.M.,M.T., IPU, ASEAN.Eng
Sekretaris Program Studi Doktor Sistem Informasi
Universitas Diponegoro

KATA SAMBUATAN II

Puji syukur kita panjatkan kepada Tuhan Yang Maha Esa atas terselesaikannya buku berjudul "Membangun Aplikasi yang Aman: Secure Software Development Lifecycle (SSDL)". Buku ini merupakan kontribusi yang sangat berarti bagi dunia pendidikan dan praktisi teknologi informasi, khususnya dalam bidang pengembangan perangkat lunak yang aman.

Topik mengenai keamanan perangkat lunak bukan hanya relevan, tetapi juga menjadi salah satu isu paling krusial dalam menghadapi tantangan teknologi masa kini. Buku ini hadir di saat yang tepat, memberikan wawasan mendalam yang dibutuhkan oleh para pengembang aplikasi, akademisi, dan praktisi keamanan siber.

Saya percaya, buku ini akan menjadi referensi penting yang tidak hanya membantu pembaca memahami konsep Secure Software Development Lifecycle, tetapi juga mempraktikkannya dalam berbagai proyek teknologi. Harapan saya, buku ini dapat menginspirasi lebih banyak inovasi dan kesadaran akan pentingnya keamanan dalam pengembangan perangkat lunak.

Akhir kata, saya mengucapkan selamat kepada penulis atas capaian luar biasa ini. Semoga buku ini dapat memberikan manfaat yang luas dan berkontribusi nyata dalam membangun ekosistem teknologi yang lebih aman dan terpercaya.

Jatmiko Endro Suseno, S.Si., M.Si., Ph.D
Co-Promotor Doktor Sistem Informasi
Universitas Diponegoro

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, buku ini akhirnya dapat diselesaikan. "Membangun Aplikasi yang Aman: Secure Software Development Lifecycle (SSDL)" merupakan hasil dari pengamatan, penelitian, dan pengalaman saya di dunia pengembangan perangkat lunak yang semakin kompleks dan menantang.

Di era digital saat ini, keamanan aplikasi menjadi salah satu aspek yang tidak bisa diabaikan. Dengan meningkatnya jumlah serangan siber dan kebocoran data, penting bagi setiap pengembang untuk memahami dan menerapkan prinsip-prinsip pengembangan perangkat lunak yang aman. Buku ini bertujuan untuk memberikan panduan menyeluruh tentang siklus hidup pengembangan perangkat lunak yang aman, mulai dari perencanaan hingga pemeliharaan.

Saya berharap buku ini bisa menjadi sumber referensi yang bermanfaat bagi para pengembang perangkat lunak, mahasiswa, dan siapa pun yang tertarik untuk memahami lebih dalam tentang keamanan dalam pengembangan aplikasi. Semoga dengan membaca buku ini, pembaca dapat lebih menyadari pentingnya keamanan dan mampu menerapkannya dalam setiap proyek yang dikerjakan.

Terima kasih kepada semua pihak yang telah mendukung saya dalam penulisan buku ini. Semoga buku ini bisa memberikan kontribusi positif bagi dunia pengembangan perangkat lunak di Indonesia.

Selamat Membaca!

Penulis

DAFTAR ISI

PERSIAPAN

Buku ini tidak hanya berisi teori, tetapi juga memberikan praktik yang mengharuskan pembaca untuk menggunakan beberapa tools. Oleh karena itu, penulis memberikan daftar-daftar apa saja yang harus pembaca persiapkan sebelum membaca buku ini.

1. Pengetahuan Dasar
    a. Pengalaman menggunakan komputer/laptop
    b. Pengalaman menggunakan Ms. Office/PDF Reader /Web Browser
    c. Pemahaman tentang programming
    d. Pemahaman tentang Jaringan & Server
    e. Pemahaman dasar cybersecurity
2. Perangkat
    a. Komputer/Laptop
3. Aplikasi
    a. Microsoft Office (Word, Excel, Powerpoint)
    b. PDF Reader (Adobe Reader, NitroPDF, dll)
    c. Browser (Firefox, Chrome, Opera, dll)
    d. XAMPP Control Panel v3.3.0
    e. ChatGPT
    f. Visual Studio Code
    g. OWASP ZAP
    h. SonarQube (Community Edition)
    i. Burp Suite (Community Edition)

Setelah mempersiapkan semuanya, kini saatnya pembaca memulai pengalaman baru yang seru dan menantang. Pesan dari penulis :

"Internet memberikan segudang informasi. Gunakanlah dengan bijak dan optimal. Dan sabar adalah kunci dalam pembelajaran yang terbaik"

Doddy Ferdiansyah, dkk - Penulis

CHAPTER 1
KONSEP SECURE SOFTWARE

Materi:

- Pengenalan SDLC dan Secure SDLC.
- Prinsip utama security by design dan security by default.
- Mengapa keamanan perlu dimulai dari awal proses pengembangan.

Durasi:

- 1 Jam

Tujuan:

- Mempelajari dan memahami dasar dan tujuan dari Secure Software Development Life Cycle.

Target Pembaca:

- Karyawan IT (SysAdmin, Network Engineer, DevOps)

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

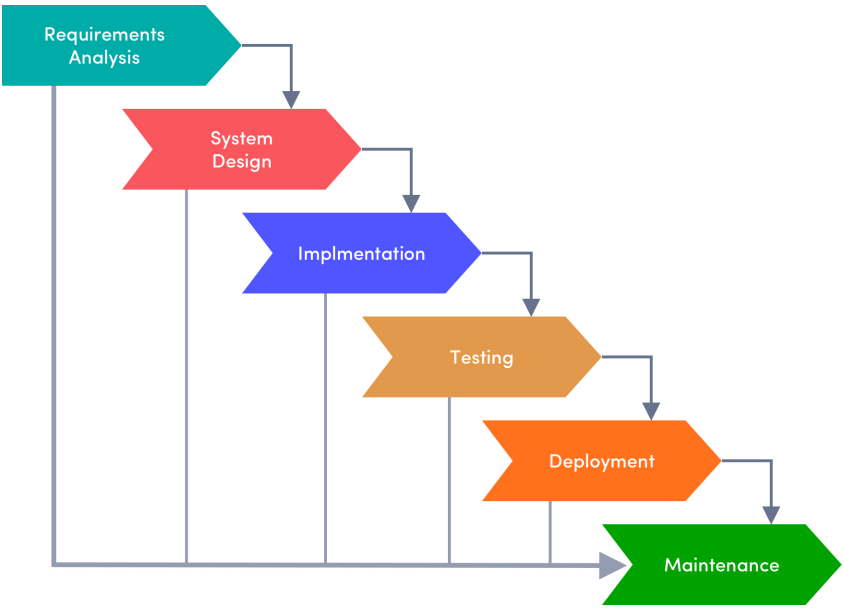
1.1. Pengenalan SDLC

SDLC adalah proses yang sistematis untuk membangun perangkat lunak dengan mengikuti serangkaian tahapan yang terstruktur. Tahapan ini memastikan bahwa perangkat lunak dikembangkan dengan cara yang efektif dan efisien. Tujuan dari Sistem Pengembangan Perangkat Lunak (SDLC) adalah untuk membangun perangkat lunak berkualitas tinggi yang tidak hanya memenuhi kebutuhan pengguna, tetapi juga dilakukan dalam batasan waktu dan anggaran yang telah ditentukan. Dengan mengikuti tahapan SDLC yang terstruktur, tim pengembang dapat meminimalkan risiko, meningkatkan efisiensi, dan menghasilkan produk yang dapat diandalkan. Selain itu, SDLC memberikan kerangka kerja yang jelas bagi seluruh tim, sehingga setiap anggota dapat memahami peran dan tanggung jawab mereka dalam proses pengembangan.

Proses ini juga memungkinkan untuk melakukan pengujian secara sistematis, yang sangat penting untuk memastikan bahwa perangkat lunak bebas dari bug dan kesalahan. Dengan pendekatan yang sistematis ini, umpan balik dari pengguna dapat diintegrasikan secara berkelanjutan, yang pada gilirannya meningkatkan kepuasan pengguna dan memperkuat posisi produk di pasar.

Lebih jauh lagi, penerapan SDLC yang baik dapat membantu organisasi dalam mengelola sumber daya dengan lebih efektif, serta mempercepat waktu peluncuran produk ke pasar, sehingga dapat memberikan keuntungan kompetitif yang signifikan.

Setiap SDLC umumnya melibatkan enam tahap utama yang saling terkait dan membentuk fondasi yang kuat untuk pengembangan perangkat lunak. Berikut tahapan-tahapan dari SDLC :



Gambar 1.1 Tahapan SDLC

Sumber :https://blog.tara.ai/wp-content/uploads/2022/05/63407-waterfall.jpg?w=1024&h=730

- Requirement Gathering: Pada tahap ini, tim pengembang bekerja sama dengan pemangku kepentingan untuk mengidentifikasi dan mendokumentasikan kebutuhan perangkat lunak. Proses ini penting untuk memastikan bahwa semua harapan dan kebutuhan pengguna terakomodasi dengan baik, sehingga perangkat lunak yang dihasilkan dapat memenuhi ekspektasi mereka.
- Design: Setelah kebutuhan teridentifikasi, tahap berikutnya adalah merancang arsitektur perangkat lunak. Di sini, pengembang menyusun blueprint atau rencana yang menggambarkan bagaimana perangkat lunak akan berfungsi, termasuk antarmuka pengguna, struktur database, dan komponen sistem lainnya. Desain yang baik menjadi kunci untuk pengembangan yang sukses.

- Implementation: Pada tahap ini, tim mulai menulis kode berdasarkan desain yang telah disiapkan. Proses ini melibatkan pemrograman, integrasi, dan pengujian unit untuk memastikan bahwa setiap bagian dari perangkat lunak berfungsi sesuai rencana. Kualitas kode yang dihasilkan sangat menentukan kinerja dan keamanan perangkat lunak.

- Testing: Setelah implementasi, perangkat lunak perlu diuji secara menyeluruh untuk memastikan bahwa tidak ada kesalahan atau bug yang tersisa. Pengujian meliputi berbagai metode seperti pengujian fungsional, pengujian integrasi, dan pengujian sistem untuk memastikan bahwa semua fitur bekerja dengan baik dan sesuai dengan spesifikasi yang telah ditetapkan.

- Deployment: Setelah perangkat lunak dianggap stabil dan bebas dari kesalahan, tahap berikutnya adalah meluncurkannya ke lingkungan produksi. Proses ini melibatkan pengaturan server, konfigurasi jaringan, dan memastikan bahwa pengguna dapat mengakses perangkat lunak dengan lancar. Tahap ini sangat penting karena merupakan saat di mana perangkat lunak mulai digunakan oleh pengguna akhir.

- Maintenance: Setelah perangkat lunak berhasil diluncurkan, proses pemeliharaan dimulai. Ini mencakup perbaikan bug yang mungkin muncul, pembaruan fitur, dan penyesuaian untuk memenuhi perubahan kebutuhan pengguna atau teknologi. Pemeliharaan yang baik memastikan perangkat lunak tetap relevan dan berfungsi dengan baik dalam jangka panjang.

1.2. Pengenalan Secure SDLC

Secure Software Development Life Cycle (SSDL) adalah kerangka kerja yang dirancang untuk memastikan bahwa keamanan menjadi aspek integral dalam setiap fase pengembangan perangkat lunak. Pendekatan ini tidak sekadar merupakan tambahan yang diterapkan setelah sistem selesai dibangun, melainkan menjadi bagian yang menyatu dalam keseluruhan proses, mulai dari tahap perencanaan hingga pemeliharaan. Tujuannya adalah untuk menanamkan pola pikir keamanan dalam tim pengembang dan memastikan bahwa semua elemen desain, pengembangan, dan implementasi perangkat lunak dibangun dengan pertimbangan keamanan yang kokoh.

Dalam paradigma Secure SDLC, setiap tahap dalam siklus hidup perangkat lunak—termasuk perencanaan, analisis, desain, implementasi, pengujian, deployment, dan pemeliharaan—dievaluasi melalui perspektif keamanan. Tujuan utamanya adalah untuk mengidentifikasi potensi ancaman dan kerentanan sedini mungkin dalam proses pengembangan. Dengan menerapkan langkah-langkah keamanan yang sesuai sebelum ancaman muncul, organisasi dapat mengurangi risiko yang dapat merusak, baik dari sisi data maupun operasional.

Pada tahap perencanaan, fokus diberikan untuk memahami kebutuhan keamanan pengguna dan bisnis serta mengekspresikan risiko yang mungkin timbul. Tahap analisis berlanjut dengan memperdalam kebutuhan ini dan mengembangkan kasus penggunaan serta skenario ancaman. Fase desain mencakup integrasi pola desain yang aman untuk membatasi risiko yang telah diidentifikasi. Implementasi melibatkan praktik pengkodean yang aman untuk mencegah kelemahan keamanan saat menulis kode. Setelah perangkat lunak diimplementasikan, pengujian

keamanan dilakukan untuk memastikan bahwa semua kebijakan dan prosedur keamanan telah dipatuhi. Deployment dilakukan dengan konfigurasi yang aman untuk melindungi sistem dari akses yang tidak sah. Tahap terakhir, pemeliharaan, tetap sangat penting karena berfokus pada pembaruan berkelanjutan dan penilaian keamanan terhadap perangkat lunak sepanjang siklus hidupnya.

Dengan mengadopsi pendekatan Secure SDLC, organisasi tidak hanya meningkatkan kualitas dan keamanan produk akhir mereka, tetapi juga dapat menurunkan biaya terkait perbaikan kerentanan setelah produk diluncurkan. Pendekatan ini berpotensi menghemat sumber daya dan waktu, sekaligus meningkatkan reputasi perusahaan dengan menyediakan produk perangkat lunak yang aman dan andal kepada pelanggan.

1.3. Manfaat Secure SDLC

Manfaat Secure SDLC mencakup peningkatan keamanan perangkat lunak sejak tahap pengembangan awal, yang membantu mencegah kerentanan dan ancaman keamanan.:

- Pengurangan Risiko Keamanan: Identifikasi masalah keamanan sejak awal untuk mencegah risiko besar saat aplikasi sudah diluncurkan.
- Efisiensi Biaya: Lebih murah memperbaiki kerentanan di awal daripada mengatasi masalah keamanan setelah aplikasi aktif.
- Kepercayaan Pengguna: Produk yang aman meningkatkan kepercayaan pengguna dan reputasi perusahaan.

Berbicara mengenai perbandingan antara SDLC dan Secure SDLC, penting untuk memahami beberapa perbedaan inti yang memengaruhi cara perangkat lunak dikembangkan dan

diimplementasikan. Dengan membandingkan kedua pendekatan ini, kita dapat melihat betapa signifikannya peran keamanan dalam setiap tahap pengembangan. Tabel perbandingan akan memberikan gambaran jelas yang menyentuh aspek utama seperti fokus proses, langkah-langkah keamanan, dan dampak pada pengeluaran biaya dan waktu.

Tabel 1.1 Perbedaan antara SDLC dengan Secure SDLC

SDLC Tradisional	Secure SDLC
Fokus pada pengembangan, fungsi, dan kualitas.	Mengutamakan keamanan sebagai bagian integral dari setiap fase.
Keamanan seringkali dilakukan sebagai langkah akhir (testing atau setelah deployment).	Melibatkan proses seperti threat modeling, risk assessment, dan secure code review di sepanjang SDLC.
Rentan terhadap serangan karena potensi kerentanan tidak diatasi sejak awal.	Mencegah kerentanan dari awal dan memastikan aplikasi dirancang dengan pendekatan keamanan yang lebih menyeluruh.

Dengan memahami perbandingan ini, diharapkan kita bisa melihat betapa pentingnya mengadopsi pendekatan Secure SDLC dalam pengembangan perangkat lunak modern.

Dengan mengintegrasikan keamanan dalam setiap tahap siklus hidup perangkat lunak, kita tidak hanya melindungi sistem kita dari ancaman, tetapi juga memberikan jaminan kepada pengguna tentang keamanan dan integritas aplikasi yang mereka gunakan.

1.4. Tahapan Secure SDLC

Setiap fase dalam Software Development Life Cycle (SDLC) dalam kerangka Secure SDLC tidak hanya menjalankan aktivitas standar yang biasa dilakukan dalam pengembangan perangkat lunak, tetapi juga dilengkapi dengan serangkaian tambahan aktivitas yang secara khusus berfokus pada aspek keamanan. Dalam tahap perencanaan, misalnya, dilakukan analisis risiko keamanan untuk memahami potensi ancaman spesifik terhadap proyek yang sedang dirancang. Tahap ini juga melibatkan penyusunan strategi keamanan yang komprehensif yang akan memandu seluruh siklus hidup pengembangan.

Gambar 1.2 Tahapan Secure SDLC

Sumber : https://blog.convisoappsec.com/en/secure-software-development-lifecycle-s-sdlc-what-is-it/

Dalam tahap Requirement Gathering, kebutuhan keamanan diidentifikasi dengan seksama. Ini mencakup aspek-aspek penting seperti otentikasi, yang memastikan bahwa hanya pengguna yang terverifikasi yang dapat mengakses sistem; otorisasi, yang menentukan hak akses pengguna; serta enkripsi, yang melindungi data dalam transit maupun data at rest dari akses yang tidak sah. Proses ini memastikan bahwa semua persyaratan keamanan diintegrasikan sejak langkah pertama pengembangan.

Pada tahap Design, dilakukan kegiatan threat modeling untuk memvisualisasikan dan memahami potensi ancaman dari

perspektif penyerang. Analisis permukaan serangan dilakukan untuk mengidentifikasi semua titik masuk potensial yang bisa dieksploitasi. Bersamaan dengan itu, dirancanglah kontrol keamanan dalam arsitektur sistem, sehingga setiap lapisan dan komponen memiliki pertahanan yang memadai terhadap ancaman.

Tahap Implementation menekankan praktik secure coding yang ketat. Validasi input adalah langkah awal krusial untuk mencegah data berbahaya masuk ke sistem. Encoding output memastikan data yang ditampilkan aman dari serangan seperti cross-site scripting (XSS). Penggunaan library yang aman dan terbukti melalui komunitas atau vendor yang terpercaya membantu mengurangi risiko yang disebabkan oleh isu keamanan yang dikenal.

Dalam tahap Testing, berbagai teknik seperti static analysis, yang memeriksa kode sumber tanpa menjalankannya, dan dynamic analysis, yang menganalisis perilaku aplikasi dalam kondisi runtime, digunakan untuk mendeteksi kerentanan secara menyeluruh. Penetration testing dilakukan untuk mensimulasikan serangan dunia nyata dan menemukan kelemahan yang mungkin terlewatkan.

Pada fase Deployment, proses penggelaran aplikasi diamankan dengan menerapkan konfigurasi server yang tepat dan aman untuk mencegah akses tidak sah serta memastikan kebijakan keamanan sistematik diimplementasikan. Penggunaan pipeline Continuous Integration/Continuous Deployment (CI/CD) yang aman sangat penting untuk mengotomatiskan proses sambil mengurangi risiko kesalahan manusia yang dapat menyebabkan kerentanan keamanan.

Tahap Maintenance tidak hanya fokus pada pembaruan rutin tetapi juga mengadopsi proses audit keamanan berkelanjutan. Ini

dilakukan untuk mengidentifikasi dan memperbaiki kerentanan yang mungkin muncul seiring berjalannya waktu, mengingat bahwa ancaman keamanan terus berkembang. Pendekatan proaktif semacam ini memastikan bahwa sistem tetap aman sepanjang siklus hidupnya.

Diskusi
Apa pengalaman anda dalam menghadapi masalah keamanan pada proyek pengembangan perangkat lunak selama ini?

1.5. Prinsip Utama Security – Security By Design

Security by Design merupakan pendekatan revolusioner dalam pengembangan perangkat lunak yang menempatkan keamanan sebagai prioritas sejak awal proses desain. Dengan integrasi keamanan di tiap langkah awal, bukan hanya sebagai tambahan di akhir, pendekatan ini menjadikan keamanan bagian esensial dari struktur dasar sistem. Hal ini memastikan bahwa sejak konsepsi, setiap fitur dan fungsi dirancang dengan pertahanan yang kuat terhadap ancaman dan serangan siber yang potensial.

Tujuan utama dari Security by Design adalah mencegah kerentanan sejak tahap dini desain, serta mengurangi risiko melalui mekanisme keamanan yang sudah menjadi bagian inherent dari arsitektur sistem. Ini tidak sekadar tentang menambahkan lapisan keamanan lebih belakangan, tetapi

membangun sistem dari pondasi dengan keamanan sebagai landasan utama.

Pendekatan utama dalam Security by Design melibatkan beberapa strategi kritis. Berikut beberapa strategi yang dapat diterapkan dalam Security by Design:

- Threat Modeling: Threat Modeling dilakukan untuk mengenali dan memahami potensi ancaman sebelum sistem dibangun, sehingga memungkinkan pengembang untuk merancang mitigasi pada awal pengembangan.
- Defense in Depth: Defense in Depth diterapkan dengan menyusun banyak lapisan keamanan pada berbagai titik dalam aplikasi, memastikan bahwa jika satu lapisan terobos, lapisan lainnya dapat memberikan perlindungan tambahan.
- Least Privilege: Prinsip Least Privilege menyarankan pemberian hak akses minimum yang diperlukan kepada pengguna atau komponen sistem, mengurangi risiko penyalahgunaan wewenang.
- Secure Defaults: Menetapkan konfigurasi default yang aman, seperti password yang kuat dan pengaturan akses yang terbatas.
- Fail-Secure: Menjamin bahwa jika sistem mengalami kegagalan, sistem tetap berada dalam keadaan aman, mencegah eksploitasi lebih lanjut.
- Input Validation and Output Encoding: Melakukan validasi input untuk mencegah serangan seperti SQL Injection dan XSS, serta mengatur keluaran agar aman.

Manfaat yang diperoleh dari Security by Design sangat signifikan. Dengan menerapkan prinsip ini, kebutuhan akan penyesuaian keamanan di tahap akhir pengembangan dapat dikurangi secara drastis. Ini tidak hanya menghemat biaya yang biasanya diperlukan untuk mengatasi kerentanan yang ditemukan di kemudian hari, tetapi juga menghilangkan kebutuhan alokasi extra waktu dalam jadwal proyek. Dengan ini, organisasi dapat meluncurkan produk yang lebih aman dengan efisiensi waktu dan biaya yang lebih baik, sambil memperkuat kepercayaan pengguna terhadap keamanan sistem yang digunakan. Security by Design, oleh karena itu, merupakan pendekatan fundamental bagi setiap organisasi yang ingin mengembangkan perangkat lunak yang aman dan andal dari awal hingga akhir.

1.6. Prinsip Utama Security – Security by Default

Security by Default adalah pendekatan proaktif dalam manajemen sistem yang menekankan praktik konfigurasi otomatis untuk memastikan bahwa sistem beroperasi dalam pengaturan yang paling aman sejak instalasi awal. Dengan prinsip ini, setiap fitur keamanan utama diaktifkan secara otomatis, sehingga pengguna tidak dibebani dengan tugas tambahan untuk mengaktifkan atau menyesuaikan pengaturan yang diperlukan untuk menjaga keamanan. Ini sangat penting dalam lingkungan di mana pengguna mungkin tidak memiliki pengetahuan teknis mendalam tentang keamanan sistem.

Tujuan utama dari Security by Default adalah menutup celah risiko keamanan yang mungkin timbul dari konfigurasi yang tidak aman. Dengan menetapkan nilai-nilai default yang kuat dan aman, sistem dilindungi dari potensi ancaman yang bisa diakibatkan oleh kelalaian atau ketidaktahuan pengguna.

Pendekatan utama yang diterapkan dalam prinsip ini melibatkan beberapa strategi seperti berikut:

- Konfigurasi Awal yang Aman: Mengaktifkan semua pengaturan keamanan penting secara default, seperti enkripsi data dan firewall, tanpa harus diaktifkan manual oleh pengguna.
- Meminimalkan Akses Terbuka: Mengaktifkan akses hanya ke layanan yang diperlukan. Misalnya, hanya port yang dibutuhkan aplikasi yang terbuka, sedangkan port lainnya tertutup secara default.
- Disable Fitur Tidak Aman Secara Default: Fitur atau layanan yang memiliki risiko keamanan lebih tinggi dinonaktifkan dari awal. Misalnya, fitur debug atau API akses jarak jauh.
- Pengaturan Password yang Aman: Menerapkan kebijakan password yang ketat secara default (misalnya, panjang minimum, kompleksitas), agar keamanan tidak tergantung pada inisiatif pengguna untuk mengubahnya.
- Enkripsi Default: Data penting seperti kredensial pengguna dan komunikasi data sudah dienkripsi secara default, sehingga tetap aman bahkan tanpa konfigurasi tambahan.

Manfaat dari pendekatan Security by Default sangatlah signifikan. Salah satunya adalah pengurangan risiko keamanan akibat kelalaian pengguna dalam mengkonfigurasi sistem dengan benar. Pengguna, terutama mereka yang mungkin tidak memiliki pemahaman keamanan yang mendalam, dapat menikmati pengalaman yang lebih aman tanpa perlu khawatir tentang pengaturan manual yang sulit atau teknis. Selain itu, dengan

meminimalkan attack surface sejak awal, potensi serangan berkurang drastis, sehingga menciptakan sistem yang lebih tahan terhadap upaya kompromi dan ancaman siber.

Perbandingan antara Security by Design dengan Security by Default memberikan wawasan yang penting dalam memahami pendekatan keamanan dalam pengembangan perangkat lunak dan sistem.

Security by Design adalah pendekatan di mana keamanan diintegrasikan ke dalam setiap tahap siklus pengembangan. Ini berarti bahwa sejak awal, sistem dirancang dengan mempertimbangkan ancaman dan risiko potensial. Tim pengembang dan desainer bekerja sama untuk memastikan bahwa setiap elemen dari arsitektur sistem dirancang untuk meminimalkan kerentanan. Keuntungan dari pendekatan ini adalah memungkinkan identifikasi dan mitigasi risiko sejak dini, yang sering kali lebih efisien dan efektif daripada perbaikan setelah sistem diterapkan.

Di sisi lain, Security by Default menekankan pada pengaturan sistem yang aman secara otomatis tanpa memerlukan intervensi pengguna. Konfigurasi default disetel untuk memberikan tingkat keamanan maksimum, mengurangi kemungkinan kesalahan pengguna yang dapat menyebabkan kerentanan. Pendekatan ini sangat penting dalam lingkungan di mana pengguna mungkin tidak memiliki pengetahuan teknis untuk melakukan konfigurasi keamanan yang kuat. Security by Default memastikan bahwa bahkan tanpa tindakan tambahan, sistem tetap terlindungi dari ancaman umum. Berikut Tabel perbandingan antara Security by Design dengan Security by Default :

Tabel 1.2 Perbandingan Security by Design dan Security by Default

Aspek	Security by Design	Security by Default
Fokus Utama	Membangun keamanan sejak awal dalam desain sistem	Mengaktifkan konfigurasi keamanan default yang aman
Pendekatan	Mengintegrasikan keamanan ke dalam arsitektur dan proses pengembangan	Menyediakan pengaturan default yang aman
Penerapan	Digunakan dalam seluruh proses pengembangan perangkat lunak	Digunakan saat pengaturan awal atau instalasi aplikasi
Keuntungan	Menghindari kerentanan dari tahap desain, mengurangi biaya perbaikan	Mengurangi risiko dari kesalahan konfigurasi pengguna
Contoh	Threat modeling, input validation	Enkripsi data default, akses terbatas secara default

Kedua pendekatan ini saling melengkapi dan sering kali digunakan bersamaan untuk mencapai tingkat keamanan yang lebih tinggi. Dengan mengimplementasikan Security by Design, organisasi dapat memastikan bahwa keamanan adalah prioritas

utama dalam pengembangan, sementara Security by Default memastikan bahwa sistem tetap aman selama digunakan.

Studi Kasus: Aplikasi Perbankan Online

Menurut anda, bagaimana konsep security by design dan security by default diterapkan pada sebuah aplikasi bank online?

1.7. Early Secure

Keamanan yang diterapkan sejak awal dalam pengembangan perangkat lunak, dikenal dengan istilah Secure by Design, adalah pendekatan yang memastikan keamanan bukan hanya sekadar tambahan di akhir, tetapi menjadi bagian dari setiap fase dalam Software Development Life Cycle (SDLC). Langkah ini memberikan berbagai manfaat, baik untuk kualitas produk maupun efisiensi dalam proses pengembangan.

1.7.1. Mengurangi Biaya Perbaikan Kerentanan

- Identifikasi Dini: Semakin cepat kerentanan atau risiko keamanan diidentifikasi, semakin mudah dan murah untuk diperbaiki. Perbaikan di tahap awal pengembangan seperti pada desain atau implementasi, secara signifikan menghemat waktu dan biaya dibandingkan perbaikan setelah perangkat lunak diluncurkan.

- Data dari NIST: Penelitian menunjukkan bahwa perbaikan kerentanan di tahap produksi dapat menjadi 10 hingga 100 kali lebih mahal dibandingkan dengan perbaikan yang dilakukan di tahap desain.

1.7.2. Mengurangi Risiko Keamanan dalam Jangka Panjang

- Keamanan yang Terintegrasi: Saat keamanan diterapkan di setiap tahap SDLC (requirement, design, implementation, testing, deployment, dan maintenance), produk yang dihasilkan akan lebih kuat dalam menghadapi ancaman keamanan.

- Pertahanan Berlapis: Dengan menambahkan mekanisme keamanan sejak awal, pengembang dapat membangun sistem yang memiliki beberapa lapisan perlindungan, sehingga lebih sulit ditembus meskipun salah satu lapisan keamanan berhasil dilanggar.

1.7.3. Meminimalkan Potensi Serangan Siber & Reputasi

- Kepercayaan Pengguna: Jika aplikasi mengalami pelanggaran data, hal ini dapat merusak reputasi organisasi dan mengurangi kepercayaan pengguna. Dengan mengamankan aplikasi sejak awal, perusahaan dapat meminimalkan risiko pelanggaran data.

- Kepatuhan Terhadap Regulasi: Banyak peraturan dan standar industri (seperti GDPR, HIPAA, dan PCI-DSS) menuntut keamanan data pengguna dan sistem. Mengintegrasikan keamanan di awal proses pengembangan membantu organisasi untuk mematuhi standar ini, menghindari potensi denda dan dampak hukum.

1.7.4. Mengurangi Kompleksitas & Memudahkan Develop

- Desain yang Aman: Dengan memperhitungkan keamanan saat merancang sistem, pengembang dapat membuat sistem yang lebih sederhana dan lebih mudah dipertahankan. Integrasi keamanan di awal menghindari tumpukan lapisan keamanan tambahan yang bisa memperumit sistem.

- Standar Secure Coding: Saat keamanan diterapkan di awal, pengembang terbiasa dengan praktik secure coding yang membantu mengurangi risiko kesalahan coding, serta menghasilkan kode yang lebih bersih dan aman.

1.7.5. Mendukung Pengembangan Terhadap Ancaman Baru

- Adaptasi terhadap Ancaman Baru: Dengan memiliki keamanan yang melekat di seluruh SDLC, pengembang lebih siap untuk merespons perubahan dalam lanskap ancaman. Sistem yang dirancang dengan mempertimbangkan keamanan lebih mudah diperbarui dan diperkuat saat ancaman baru muncul.

- Threat Modeling: Di tahap awal, threat modeling dapat diterapkan untuk mengidentifikasi ancaman yang mungkin muncul. Ini membuat tim lebih siap untuk mengantisipasi potensi serangan baru dan melakukan perubahan desain yang diperlukan.

1.7.6. Perbandingan Keamanan Awal vs. Keamanan Akhir

Sebagai bagian penting dari manajemen risiko dalam teknologi informasi, keamanan awal dan keamanan akhir memiliki peran yang berbeda dalam melindungi sistem dan data.

Keamanan awal melibatkan penerapan langkah-langkah keamanan sejak tahap perancangan dan pengembangan sistem. Ini

termasuk identifikasi potensi kerentanan dan ancaman sebelum sistem diimplementasikan. Dengan melakukan ini, organisasi dapat mengurangi risiko serangan dan kebocoran data sejak dini. Contoh dari pendekatan ini adalah integrasi praktik pengkodean aman dan penilaian risiko selama tahap desain.

Di sisi lain, keamanan akhir berfokus pada penerapan langkah-langkah keamanan setelah sistem sudah berjalan. Ini mencakup pemantauan berkelanjutan, deteksi ancaman, dan respons terhadap insiden yang terjadi. Keamanan akhir penting untuk memastikan sistem tetap aman dari ancaman yang terus berkembang dan untuk menanggapi serangan yang mungkin terjadi. Metode seperti patching rutin, audit keamanan, dan penggunaan alat deteksi intrusi merupakan bagian dari pendekatan ini.

Meskipun kedua pendekatan ini memiliki fokus yang berbeda, keduanya saling melengkapi dalam menjaga keamanan sistem secara keseluruhan. Pendekatan keamanan yang efektif biasanya menggabungkan elemen dari kedua strategi ini untuk memastikan perlindungan maksimal dari ancaman yang ada.

Berikut tabel perbandingan Pendekatan Keamanan Awal dengan Keamanan Akhir:

Tabel 1.3 Perbandingan Keamanan Awal vs Keamanan Akhir

Aspek	Keamanan Awal dalam SDLC	Keamanan di Akhir SDLC
Biaya	Rendah, karena perbaikan dini	Tinggi, karena perubahan besar mungkin diperlukan

Efektivitas	Lebih efektif, karena diterapkan di setiap tahap	Kurang efektif, keamanan sebagai tambahan
Risiko Kerentanan	Lebih rendah, karena kerentanan diidentifikasi dini	Lebih tinggi, kerentanan sering ditemukan di akhir
Kepatuhan Regulasi	Memungkinkan kepatuhan penuh	Berpotensi melanggar jika ada kerentanan yang belum diperbaiki
Dampak Terhadap Pengguna	Tingkat keamanan lebih baik dan konsisten	Rentan terhadap serangan setelah peluncuran

1.8. Diskusi & Studi Kasus

Ancaman Umum yang Sering Diabaikan di Setiap Tahap SDLC pada :

- Requirement Gathering
- Design
- Implementation
- Testing
- Deployment
- Maintenance

Contoh Kasus Diskusi

- Kasus: Sebuah aplikasi fintech diluncurkan tanpa pengujian keamanan, dan setelah beberapa bulan,

ditemukan bahwa aplikasi tersebut rentan terhadap serangan SQL Injection dan XSS. Data pengguna berisiko tinggi terekspos, dan tim tidak memiliki logging yang baik untuk melacak serangan.

- Tugas: Identifikasi tahap mana yang mungkin mengabaikan praktik keamanan, dan solusi apa yang seharusnya diterapkan sejak awal.

CHAPTER 2
SECURE SOFTWARE REQUIREMENT

Materi:

- Teknik pengumpulan kebutuhan keamanan.
- Menetapkan kebutuhan keamanan fungsional dan non-fungsional.
- Mengidentifikasi ancaman (misalnya, STRIDE model).

Durasi:

- 1 Jam

Tujuan:

- Menentukan dan mendokumentasikan kebutuhan keamanan.

Target Pembaca:

- Karyawan IT (SysAdmin, Network Engineer, DevOps)

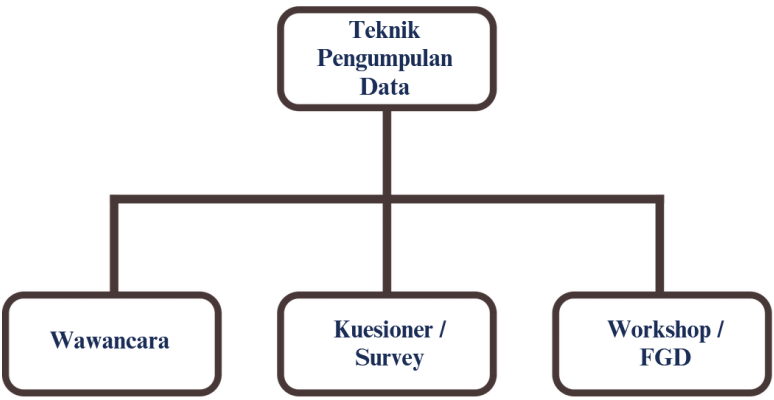
Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

2.1. Pengumpulan Data

Pada tahap pengumpulan kebutuhan keamanan, tujuan utama adalah untuk mendapatkan pemahaman yang mendalam tentang kebutuhan aplikasi dari sudut pandang keamanan. Hal ini melibatkan identifikasi dan penilaian risiko serta potensi ancaman yang mungkin timbul seiring dengan penggunaan aplikasi. Melalui analisis mendetail ini, tim pengembang dapat memahami berbagai jenis ancaman yang dapat dihadapi oleh aplikasi, baik dari faktor eksternal seperti serangan siber maupun dari faktor internal seperti kesalahan pengguna atau kebijakan keamanan yang kurang memadai. Pendekatan ini sangat penting karena memastikan bahwa aspek keamanan bukanlah sekadar tambahan atau setelah-pikiran, tetapi menjadi bagian integral yang tidak terpisahkan dalam sistem yang akan dikembangkan.

Dengan menganalisis kebutuhan keamanan secara proaktif, tim dapat merancang solusi yang lebih efektif dan responsif terhadap ancaman potensial yang teridentifikasi. Ini juga memungkinkan tim untuk menyusun strategi mitigasi yang efektif, menetapkan kontrol keamanan yang diperlukan, dan memastikan bahwa semua persyaratan keamanan pengguna tercakup dalam desain sistem.



Gambar 2.1 Teknik Pengumpulan Data (Sumber: Penulis)

Selanjutnya, pengumpulan informasi ini mencakup kolaborasi dengan berbagai pemangku kepentingan, termasuk pengguna akhir, analis bisnis, dan tim keamanan untuk mengumpulkan pandangan dan perhatian yang beragam terhadap keamanan aplikasi. Dengan demikian, hasil dari fase ini memberikan fondasi yang kuat untuk langkah-langkah desain dan implementasi berikutnya, sekaligus membangun kesadaran dan komitmen terhadap keamanan di seluruh siklus hidup pengembangan perangkat lunak. Melalui pendekatan yang holistik dan menyeluruh ini, organisasi dapat memperkuat postur keamanan aplikasi mereka dan melindungi data serta aset penting dari ancaman yang ada. Untuk teknik pengumpulan informasi terdiri dari 3 teknik utama yaitu wawancara, kuesioner atau survey, dan Forum Grup Diskusi. Masing-masing teknik ini memiliki kelebihan dan kekurangan yang dapat disesuaikan dengan kebutuhan. Berikut penjelasan dari masing-masing teknik pengumpulan data:

Wawancara (Interviews)

- Deskripsi: Teknik ini melibatkan sesi tanya-jawab langsung dengan berbagai pemangku kepentingan (stakeholders) seperti pengguna akhir, pemilik sistem, dan tim keamanan.
- Tujuan: Mendapatkan wawasan langsung mengenai kebutuhan dan ekspektasi terkait keamanan dari sudut pandang pengguna atau bisnis.
- Proses:
    - Persiapkan daftar pertanyaan yang difokuskan pada area yang rentan, seperti akses data, perlindungan data sensitif, dan potensi serangan.
    - Ajak stakeholder mendiskusikan skenario-skenario di mana mereka merasa ada kebutuhan perlindungan ekstra.

Kuesioner dan Survei

- Deskripsi: Menggunakan kuesioner untuk mendapatkan jawaban dari sejumlah besar orang mengenai kebutuhan keamanan sistem.
- Tujuan: Menstandarisasi tanggapan dari pemangku kepentingan mengenai risiko, kebutuhan akses, dan prioritas keamanan.
- Proses:
    - Buat kuesioner yang terstruktur dengan pertanyaan terkait kebutuhan keamanan dan risiko.
    - Distribusikan ke pemangku kepentingan dan analisis hasilnya untuk mengidentifikasi pola kebutuhan.

Workshops dan Focus Groups

- Deskripsi: Teknik ini mengumpulkan pemangku kepentingan dari berbagai fungsi untuk berdiskusi dalam kelompok kecil, memungkinkan eksplorasi kebutuhan keamanan secara lebih mendalam.
- Tujuan: Mendorong kolaborasi dan menciptakan kesepahaman bersama mengenai kebutuhan keamanan.
- Proses:
    - Undang perwakilan dari berbagai tim (teknis, bisnis, legal, dll.).
    - Diskusikan ancaman potensial, skenario risiko, dan kebutuhan keamanan yang mungkin diabaikan dalam pengembangan.
    - Gunakan teknik brainstorming untuk mengidentifikasi kebutuhan keamanan.

2.2. Fungsional & Non-Fungsional

Dalam proses pengembangan perangkat lunak, memahami kebutuhan pengguna adalah langkah krusial yang mempengaruhi keberhasilan proyek. Salah satu cara untuk mengelompokkan kebutuhan tersebut adalah dengan membaginya menjadi dua kategori utama: kebutuhan fungsional dan kebutuhan non-fungsional. Kedua kategori ini memiliki peran yang berbeda tetapi sama pentingnya dalam memastikan perangkat lunak dapat memenuhi harapan pengguna dan berfungsi dengan baik.

Kebutuhan Keamanan Fungsional adalah kebutuhan yang secara langsung terkait dengan fungsi yang disediakan oleh sistem untuk menjaga keamanannya. Fungsionalitas ini biasanya mencakup fitur keamanan yang bisa dilihat dan berinteraksi langsung oleh pengguna atau sistem. Kebutuhan Keamanan Non-Fungsional adalah aspek keamanan yang tidak terlihat oleh pengguna secara langsung namun penting untuk memastikan aplikasi berfungsi dengan aman. Ini mencakup ketahanan terhadap ancaman, ketersediaan, ketepatan waktu dalam menanggapi insiden keamanan, dan kepatuhan terhadap standar. Selain itu, baik kebutuhan keamanan fungsional maupun non-fungsional harus dirancang dengan cermat agar dapat bekerja secara harmonis. Keduanya harus dievaluasi secara berkala untuk memastikan bahwa sistem tetap terlindungi dari ancaman yang terus berkembang. Pengujian keamanan yang rutin dan audit juga merupakan bagian penting dari pemeliharaan keamanan sistem, memastikan bahwa setiap celah atau kelemahan dapat diidentifikasi dan ditangani sebelum dapat dieksploitasi oleh pihak yang tidak bertanggung jawab. Dengan pendekatan yang komprehensif ini, organisasi dapat mempertahankan kepercayaan pengguna dan menjaga integritas serta kerahasiaan data yang dikelola.

2.2.1. Cakupan Kebutuhan Keamanan Fungsional

Kebutuhan keamanan fungsional mencakup:

- Otentikasi Pengguna: Mekanisme untuk mengonfirmasi identitas pengguna, seperti username dan password atau otentikasi multifaktor.
- Kontrol Akses: Kontrol akses berbasis peran untuk membatasi akses pengguna sesuai peran mereka.
- Enkripsi Data: Enkripsi data sensitif saat disimpan dan dikirimkan.
- Pencatatan Aktivitas (Logging): Mencatat aktivitas pengguna untuk audit dan deteksi anomali.
- Pemulihan Kata Sandi yang Aman: Proses yang melibatkan konfirmasi melalui email atau telepon untuk mencegah akses tidak sah.

2.2.2. Cakupan Kebutuhan Keamanan Non-Fungsional

Kebutuhan keamanan non-fungsional mencakup kualitas yang menjaga sistem tetap aman, antara lain:

- Ketahanan terhadap serangan (Resilience) untuk menghadapi serangan tanpa menghentikan layanan.
- Ketersediaan (Availability) dengan tingkat uptime tinggi bagi pengguna.
- Performa keamanan (Security Performance) agar fitur keamanan tidak mengganggu performa aplikasi.
- Keberlanjutan logging dan pemantauan (Auditability) untuk penyimpanan log sesuai kebutuhan regulasi.
- Pemulihan bencana (Disaster Recovery) dengan rencana pemulihan data setelah insiden.

2.2.3. Contoh Penerapan Kebutuhan Fungsional

Penerapan kebutuhan fungsional dalam pengembangan sistem perangkat lunak sangat krusial bagi perusahaan yang ingin tetap kompetitif dan memenuhi harapan pengguna. Kebutuhan fungsional memberikan arah yang jelas bagi tim pengembang mengenai apa yang harus dicapai oleh perangkat lunak. Dalam bagian ini, kita akan mengeksplorasi contoh penerapan kebutuhan fungsional di perusahaan, menunjukkan bagaimana kebutuhan ini dapat diterjemahkan menjadi fitur memperkuat kinerja dan efisiensi organisasi.

- Otentikasi Pengguna
    - Penerapan: Bank XYZ mengembangkan aplikasi mobile banking yang memerlukan otentikasi pengguna yang kuat. Saat pengguna pertama kali mendaftar, mereka diwajibkan untuk membuat username dan password yang unik. Setelah login pertama, pengguna diharuskan mengaktifkan otentikasi dua faktor (2FA), di mana mereka harus memasukkan kode yang diterima melalui SMS atau aplikasi autentikasi setiap kali mereka login. Ini memastikan bahwa hanya pengguna yang sah yang dapat mengakses akun mereka.
- Pengelolaan Akses
    - Penerapan: Perusahaan Software ABC memiliki platform manajemen proyek yang digunakan oleh karyawan di berbagai departemen. Dalam sistem ini, pengelolaan akses dilakukan melalui kontrol akses berbasis peran (RBAC). Misalnya, anggota tim pengembang memiliki akses untuk melihat dan mengedit tugas pengembangan, sementara anggota

tim pemasaran hanya dapat melihat status proyek tanpa kemampuan untuk mengubahnya. Admin sistem memiliki hak akses penuh untuk mengelola dan mengubah semua pengaturan dan peran. Dengan cara ini, perusahaan dapat memastikan bahwa hanya orang yang tepat yang memiliki akses ke informasi sensitif.

- Audit dan Logging Penelusuran
    - Penerapan: Perusahaan E-Commerce DEF menerapkan fitur logging yang komprehensif pada platform e-commerce mereka. Semua aktivitas pengguna, seperti login, pembelian, dan pengubahan informasi profil, dicatat dengan menggunakan timestamp dan alamat IP. Data ini digunakan untuk melakukan audit rutin, sehingga jika terjadi aktivitas mencurigakan seperti upaya penipuan atau pelanggaran data, tim keamanan dapat dengan cepat mengakses log yang relevan untuk melakukan analisis dan mengidentifikasi pelaku.

2.2.4. Contoh Penerapan Kebutuhan Non-Fungsional

Dalam pengembangan sistem perangkat lunak, kebutuhan tidak hanya terbatas pada apa yang sistem harus lakukan, tetapi juga bagaimana sistem tersebut harus berfungsi. Inilah yang menjadi fokus kebutuhan non-fungsional. Kebutuhan non-fungsional mencakup aspek kualitas dan karakteristik yang menentukan seberapa baik sistem dapat beroperasi dalam lingkungan yang ditentukan. Penerapan kebutuhan non-fungsional yang tepat dapat meningkatkan pengalaman pengguna, efisiensi operasional, dan keamanan sistem secara keseluruhan. Berikut adalah beberapa

contoh penerapan kebutuhan non-fungsional di perusahaan yang berbeda.

- Keamanan
    - Penerapan: Bank ABC mengembangkan aplikasi perbankan online dan menetapkan keamanan sebagai salah satu kebutuhan non-fungsional utama. Mereka mengimplementasikan enkripsi kuat untuk data yang dikirim antara klien dan server dengan menggunakan protokol TLS (Transport Layer Security). Selain itu, seluruh data sensitif, seperti informasi rekening dan transaksi, dienkripsi saat disimpan dalam database. Bank ini juga menerapkan kebijakan untuk memantau secara real-time untuk mendeteksi dan merespons insiden keamanan, sehingga dapat mengambil tindakan cepat jika ada upaya akses yang tidak sah.
- Ketersediaan
    - Penerapan: Perusahaan E-Commerce XYZ menginginkan sistem yang memiliki tingkat ketersediaan tinggi, dengan target minimal 99,9% uptime. Mereka mencapai hal ini dengan merancang infrastruktur berbasis cloud yang memiliki pengaturan redundansi. Server dan database terdistribusi di beberapa lokasi geografis yang berbeda untuk menjaga agar situs web tetap online, bahkan jika salah satu server mengalami masalah. Selain itu, mereka menerapkan rencana pemulihan bencana yang komprehensif untuk memastikan bahwa data dan layanan dapat dipulihkan dengan cepat setelah terjadinya kegagalan.

- Kepatuhan
    - Penerapan: Perusahaan Kesehatan DEF harus mematuhi berbagai regulasi privasi data, termasuk HIPAA (Health Insurance Portability and Accountability Act). Sebagai bagian dari pengembangan sistem manajemen data pasien, mereka menerapkan kontrol keamanan dan prosedur audit untuk memastikan kepatuhan terhadap undang-undang tersebut. Ini termasuk pengaturan akses yang ketat, pelatihan staf mengenai keamanan data, dan pengujian rutin untuk memastikan bahwa perangkat lunak berfungsi dalam batasan yang ditentukan oleh regulasi. Tanpa mematuhi kepatuhan ini, perusahaan dapat menghadapi denda yang signifikan dan kehilangan kepercayaan pasien.

2.3. Identifikasi Risiko & Ancaman

Identifikasi risiko dan ancaman adalah langkah krusial dalam Secure Software Development Life Cycle (SDLC) yang bertujuan untuk memahami dan mengevaluasi potensi masalah keamanan yang dapat mempengaruhi sistem atau aplikasi yang sedang dikembangkan. Proses ini melibatkan pengumpulan informasi tentang semua aspek yang dapat memengaruhi keamanan produk akhir, mulai dari desain hingga implementasi dan pemeliharaan. Berikut adalah penjelasan mendalam tentang kegiatan ini, mengapa penting, serta teknik dan metode yang sering digunakan.

- Risiko: Risiko dalam konteks keamanan perangkat lunak adalah kemungkinan terjadinya peristiwa yang dapat membahayakan integritas, kerahasiaan, dan ketersediaan data. Risiko ini diukur berdasarkan dua faktor:

kemungkinan terjadinya peristiwa tersebut dan dampaknya jika peristiwa itu terjadi.

- Ancaman: Ancaman adalah faktor eksternal atau internal yang dapat mengeksploitasi kerentanan dalam sistem atau aplikasi. Ancaman dapat berupa serangan siber (seperti malware, phishing, atau serangan DDoS), kesalahan manusia, atau bencana alam yang dapat mempengaruhi operasional sistem.

Identifikasi risiko dan ancaman sangat penting dalam pengembangan perangkat lunak karena:

- Perlindungan Data: Dengan memahami risiko, organisasi dapat mengimplementasikan langkah-langkah untuk melindungi data sensitif dan aset penting.
- Kepatuhan: Banyak regulasi dan standar industri, seperti GDPR atau ISO 27001, mensyaratkan identifikasi dan pengelolaan risiko sebagai bagian dari proses keamanan.
- Meningkatkan Kepercayaan Pelanggan: Sistem yang aman dan dapat diandalkan meningkatkan kepercayaan pelanggan dan mitra bisnis.
- Mengurangi Biaya Potensial: Dengan mengidentifikasi risiko lebih awal, perusahaan dapat menghindari biaya terkait dengan pelanggaran data atau gangguan layanan.

Untuk mengidentifikasi ancaman keamanan yang potensial, kita dapat menggunakan dua model populer: STRIDE dan DREAD. Keduanya membantu mengevaluasi ancaman yang mungkin muncul pada setiap tahapan pengembangan perangkat lunak dan bisa membantu tim memahami potensi risiko yang perlu dikelola.

2.3.1. Model STRIDE

Model STRIDE dikembangkan oleh Microsoft untuk mengklasifikasikan jenis-jenis ancaman keamanan. Setiap huruf dalam STRIDE merepresentasikan kategori ancaman yang berbeda. Dengan menggunakan STRIDE, kita dapat mengevaluasi aplikasi dari berbagai perspektif ancaman yang mungkin muncul. Berikut adalah penjelasan STRIDE dan contoh untuk setiap kategori:

Tabel 2.1 Model STRIDE

Kategori	Definisi	Contoh
Spoofing	Upaya untuk menyamar sebagai pengguna atau entitas lain.	Penyerang mencoba masuk ke aplikasi dengan menggunakan kredensial pengguna lain.
Tampering	Memodifikasi data atau komponen aplikasi secara tidak sah.	Penyerang mengubah data transaksi atau data pengguna tanpa izin.
Repudiation	Tidak adanya bukti atas suatu tindakan yang dilakukan dalam aplikasi, sehingga sulit ditelusuri.	Seorang pengguna membantah bahwa ia pernah melakukan transaksi tertentu.
Information Disclosure	Pengungkapan data sensitif yang seharusnya terlindungi.	Aplikasi tidak melindungi data sensitif (misal: nomor kartu kredit) dan data ini

Kategori	Definisi	Contoh
		terlihat oleh pihak yang tidak berwenang.
Denial of Service	Serangan yang bertujuan menghentikan atau menurunkan performa layanan aplikasi.	Penyerang mengirimkan sejumlah besar permintaan ke server sehingga server menjadi tidak dapat diakses oleh pengguna.
Elevation of Privilege	Meningkatkan hak akses yang dimiliki oleh pengguna atau proses tertentu.	Penyerang berhasil mengubah hak akses mereka dari pengguna biasa menjadi administrator.

Misalkan kita sedang mengembangkan aplikasi e-commerce yang akan menyimpan data transaksi pengguna. Dalam tahap pengumpulan kebutuhan, kita bisa melakukan analisis STRIDE untuk mengidentifikasi ancaman sebagai berikut:

- Spoofing: Tetapkan kebutuhan untuk otentikasi yang aman, seperti Multi-Factor Authentication (MFA), untuk mencegah penyamaran identitas.
- Tampering: Tetapkan kebutuhan enkripsi data dan validasi integritas untuk mencegah perubahan data tanpa izin.
- Repudiation: Gunakan sistem pencatatan log (audit trail) untuk memastikan setiap tindakan dapat ditelusuri kembali jika terjadi masalah.

- Information Disclosure: Pastikan hanya pihak yang berwenang memiliki akses ke data sensitif, seperti informasi pembayaran dan alamat pengguna.
- Denial of Service: Identifikasi kebutuhan untuk menerapkan rate-limiting atau firewall untuk mencegah serangan DoS.
- Elevation of Privilege: Gunakan kontrol akses yang ketat untuk mencegah pengguna dengan hak akses terbatas meningkatkan hak akses mereka tanpa izin.

2.3.2. Model DREAD

DREAD adalah model penilaian risiko yang digunakan untuk menghitung seberapa besar dampak dari sebuah ancaman. Setiap huruf dalam DREAD mewakili komponen yang dinilai untuk setiap ancaman. Skor untuk tiap komponen dapat digunakan untuk menghitung total risiko dari suatu ancaman. Berikut adalah rincian dari DREAD:

Tabel 2.2 Model DREAD

Kategori	Definisi	Contoh
Damage	Tingkat kerusakan yang ditimbulkan jika ancaman terjadi.	Apakah data atau layanan akan rusak atau hilang jika terjadi ancaman?
Reproducibility	Kemudahan bagi penyerang untuk mengulangi ancaman yang sama.	Apakah serangan ini mudah untuk direplikasi? Misalnya, serangan SQL injection mungkin mudah direplikasi.

Kategori	Definisi	Contoh
Exploitability	Kemudahan bagi penyerang untuk melancarkan serangan.	Apakah penyerang hanya membutuhkan sedikit sumber daya atau pengetahuan untuk melancarkan serangan ini?
Affected Users	Jumlah pengguna atau data yang akan terdampak jika ancaman terjadi.	Apakah hanya satu pengguna yang terpengaruh atau banyak pengguna yang bisa terkena dampaknya?
Discoverability	Seberapa mudah ancaman ini ditemukan oleh penyerang.	Apakah kelemahan ini mudah terlihat atau perlu keahlian khusus untuk ditemukan?

Setiap kategori dapat diberikan skor dari 1 hingga 10, di mana 1 berarti risiko rendah dan 10 berarti risiko tinggi. Skor total akan menentukan prioritas dari ancaman yang ditemukan.

Misalkan kita memiliki ancaman SQL Injection dalam aplikasi e-commerce:

- Damage: Jika terjadi, data sensitif bisa terekspos. Skor: 8
- Reproducibility: Serangan ini bisa dilakukan berulang kali pada berbagai titik input yang rentan. Skor: 9
- Exploitability: Serangan SQL Injection mudah dilakukan dengan sedikit pengetahuan dasar. Skor: 9

- Affected Users: Jika berhasil, bisa berdampak pada banyak pengguna. Skor: 8
- Discoverability: SQL Injection biasanya mudah ditemukan dengan teknik sederhana. Skor: 9

Dengan menggunakan model DREAD, kita dapat menghitung skor risiko untuk ancaman SQL Injection dengan menjumlahkan skor dari setiap komponen yang telah diberikan:

- Damage: 8
- Reproducibility: 9
- Exploitability: 9
- Affected Users: 8
- Discoverability: 9

Perhitungan:

Total Risk Score

=Damage+Reproducibility+Exploitability+Affected Users+Discoverability

=8+9+9+8+9=43

Hasil:

Skor Risiko Total untuk SQL Injection adalah 43.

Skor ini memberikan gambaran tentang tingkat risiko yang terkait dengan ancaman SQL Injection dalam aplikasi e-commerce. Semakin tinggi skornya, semakin besar risiko yang harus dikelola atau diminimalkan melalui tindakan keamanan yang tepat.

2.4. Studi Kasus

Latihan 1 : Identifikasi Ancaman dengan Model STRIDE

Setiap kelompok fokus pada satu fitur aplikasi, misalnya:

- Fitur Pendaftaran Akun
- Fitur Pembayaran
- Fitur Keranjang Belanja
- Panel Admin

Gunakan model STRIDE untuk mengidentifikasi ancaman keamanan pada setiap fitur yang sedang dievaluasi. Setiap kelompok akan mengidentifikasi ancaman potensial untuk setiap kategori STRIDE, yaitu:

- Spoofing: Adakah risiko penyamaran identitas pada fitur ini?
- Tampering: Apakah ada potensi modifikasi data yang tidak sah?
- Repudiation: Apakah ada cara untuk menyangkal aktivitas di aplikasi tanpa bukti?
- Information Disclosure: Apakah informasi sensitif berpotensi terekspos?
- Denial of Service: Bisakah aplikasi atau fitur ini dimatikan atau diperlambat?
- Elevation of Privilege: Apakah mungkin pengguna dengan hak akses rendah dapat mengakses hak yang lebih tinggi?

Latihan 2 : Penilaian Ancaman dengan Model DREAD

Setiap kelompok memilih salah satu ancaman yang telah mereka identifikasi di STRIDE dan menggunakan model DREAD untuk menilai tingkat risiko dari ancaman tersebut.

- Damage: Seberapa besar dampak dari ancaman ini jika berhasil dilakukan?
- Reproducibility: Seberapa mudah ancaman ini dilakukan berulang kali?
- Exploitability: Seberapa mudah ancaman ini dilakukan oleh penyerang?
- Affected Users: Berapa banyak pengguna yang akan terpengaruh?
- Discoverability: Seberapa mudah ancaman ini ditemukan oleh penyerang?

Berikan Skor DREAD: Peserta memberi skor dari 1 (risiko rendah) hingga 10 (risiko tinggi) untuk setiap kategori DREAD. Lalu, hitung total skor.

CHAPTER 3
SECURE SOFTWARE DESIGN

Materi:

- Teknik pengumpulan kebutuhan keamanan.
- Menetapkan kebutuhan keamanan fungsional dan non-fungsional.
- Mengidentifikasi ancaman (misalnya, STRIDE model).

Durasi:

- 1,5 Jam

Tujuan:

- Menerapkan konsep secure design untuk mitigasi risiko keamanan.

Target Pembaca:

- Karyawan IT (SysAdmin, Network Engineer, DevOps)

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

3.1. Prinsip Security Design

Dalam era digital yang semakin kompleks dan saling terhubung, keamanan perangkat lunak dan sistem informasi menjadi prioritas utama bagi organisasi dari berbagai ukuran dan sektor. Sebagai respons terhadap meningkatnya ancaman siber, prinsip keamanan desain (Security Design Principles) muncul sebagai fondasi penting dalam pengembangan sistem yang aman. Prinsip-prinsip ini tidak hanya berfokus pada penerapan kontrol keamanan, tetapi juga pada cara mendesain aplikasi dan sistem sehingga secara intrinsik lebih tahan terhadap serangan.

Prinsip security design adalah pedoman dan strategi yang diadopsi oleh pengembang dan arsitek sistem untuk menciptakan aplikasi dan sistem yang aman. Prinsip-prinsip ini bertujuan untuk mengintegrasikan pertimbangan keamanan sejak awal dalam proses desain dan pengembangan, bukan hanya sebagai tambahan di tahap akhir. Ini membantu dalam meminimalkan potensi kerentanan dan serangan yang dapat merusak integritas, kerahasiaan, dan ketersediaan data.

Dalam konteks Secure Software Design, prinsip least privilege, defense in depth, dan fail-safe defaults sangat penting. Ketiga prinsip ini adalah dasar untuk menciptakan sistem yang aman sejak tahap desain, dan mereka membantu meminimalkan risiko jika terjadi serangan.

3.1.1. Prinsip 1: Least Privilege

Least privilege adalah prinsip di mana setiap komponen, pengguna, atau proses dalam sistem hanya memiliki hak akses minimum yang diperlukan untuk melakukan tugasnya. Prinsip ini membatasi kemampuan entitas dalam sistem untuk mencegah mereka mengakses informasi atau menjalankan fungsi yang tidak relevan dengan tugas mereka.

Penerapan dalam Sistem:

- Akses Terbatas untuk Pengguna dan Proses: Berikan hak akses terbatas pada akun pengguna atau layanan. Misalnya, dalam aplikasi keuangan, pengguna biasa tidak seharusnya memiliki akses ke data administrator.
- Kontrol Akses Berdasarkan Peran (Role-Based Access Control): Implementasikan kontrol akses yang membatasi hak istimewa setiap peran, seperti pengguna, admin, dan developer.
- Isolasi Modul atau Proses: Setiap modul dalam sistem atau proses yang berjalan harus memiliki izin yang terbatas, hanya untuk fungsi tertentu, misalnya dalam skenario microservices di mana masing-masing layanan hanya bisa mengakses data atau API yang terkait.

Keuntungan:

- Mengurangi dampak jika terjadi eksploitasi pada satu akun atau layanan.
- Membatasi kerusakan dari kesalahan manusia atau proses otomatis yang bermasalah.

Misalnya, kita mendesain sistem aplikasi keuangan yang memiliki tiga jenis pengguna utama:

- Customer – hanya memiliki akses untuk melihat saldo, melakukan transfer, dan melihat riwayat transaksi.
- Customer Service (CS) Agent – memiliki akses untuk melihat data akun pengguna dan membantu pengguna dalam memulihkan akses akun.
- Admin – memiliki akses untuk mengelola data seluruh pengguna, mengelola transaksi, dan menjalankan laporan keuangan.

Langkah-langkah Penerapan dalam Desain:

- Identifikasi Peran dan Hak Akses Minimum:
    - Daftar peran (Customer, CS Agent, Admin) dan kebutuhan masing-masing.
    - Misalnya, Customer hanya membutuhkan akses ke akun mereka sendiri, sedangkan CS Agent memerlukan akses baca ke data pelanggan tetapi tidak bisa mengubah informasi saldo atau riwayat transaksi.
- Tentukan Batasan Akses Berdasarkan Tugas:
    - Customer: Akses akun pribadi, tanpa akses ke akun lain.
    - CS Agent: Melihat data pengguna untuk membantu, tanpa akses ke transaksi sensitif.
    - Admin: Akses penuh untuk pengelolaan, tapi terbatas pada data yang relevan.
- Desain Kontrol Akses Berbasis Peran:

    - Implementasi RBAC (Role-Based Access Control), membatasi izin sesuai peran. Hanya Admin yang bisa mengubah hak akses.

- Penentuan Batasan Teknis:
    - Batasi akses API dan database; endpoint API untuk melihat riwayat transaksi hanya boleh diakses oleh Customer sendiri, dan data akun pengguna hanya dapat diakses oleh CS Agent setelah verifikasi identitas.
    - Batasi query SQL sehingga CS Agent hanya dapat melakukan pencarian atas data tertentu yang sudah difilter
- Audit dan Log Aktivitas:
    - Implementasikan logging untuk mencatat setiap akses berdasarkan peran.
- Penerapan Separation of Duties:
    - Pisahkan tugas kritis; CS Agent tidak bisa memproses transaksi, hanya Customer yang bisa. Dan admin hanya bisa memproses transaksi besar dengan otorisasi tertentu.

3.1.2. Prinsip 2: Defense in Depth

Defense in depth adalah strategi keamanan di mana beberapa lapisan pertahanan diterapkan pada sistem, sehingga jika satu lapisan dilanggar, lapisan berikutnya masih dapat melindungi sistem. Prinsip ini mengasumsikan bahwa setiap kontrol keamanan mungkin gagal, sehingga beberapa lapisan kontrol dibutuhkan untuk memitigasi risiko.

Penerapan dalam Sistem:

- Firewall dan Network Segmentation: Melindungi jaringan dengan firewall untuk memblokir akses dari luar dan membagi jaringan menjadi segmen-segmen yang lebih kecil untuk mengontrol pergerakan lateral.
- Otentikasi Multi-Faktor (MFA): Menambahkan lapisan keamanan tambahan selain password, seperti otentikasi berbasis kode OTP atau biometrik.
- Enkripsi Data: Melindungi data pada setiap lapisan penyimpanan dan pengiriman dengan enkripsi untuk menghindari kebocoran jika data disadap.
- Logging dan Monitoring: Memantau aktivitas yang mencurigakan melalui log dan analisis keamanan secara berkelanjutan.

Keuntungan:

- Memperkuat ketahanan sistem terhadap serangan yang kompleks.
- Memberikan waktu untuk merespons serangan pada lapisan-lapisan berikutnya jika satu lapisan berhasil ditembus.

Misalnya, kita merancang aplikasi web perbankan yang memungkinkan pengguna mengakses informasi rekening mereka dan melakukan transaksi. Untuk mengamankan aplikasi ini, kita dapat menerapkan beberapa lapisan keamanan pada setiap komponen sistemnya. Dengan menerapkan Defense in Depth, desain sistem mengalokasikan beberapa mekanisme keamanan pada berbagai level, membuat jalur serangan lebih sulit bagi penyerang.

Contoh berikut menggabungkan langkah-langkah di atas:

- Frontend dan Backend: Validasi input pada frontend untuk mencegah serangan langsung ke sistem, namun tetap tambahkan validasi yang lebih kuat di backend untuk menambah lapisan keamanan.
- API Management: Gunakan API Gateway untuk memeriksa token pengguna, memonitor permintaan API, dan menerapkan rate-limiting.
- Server Security: Pastikan semua server berada di belakang firewall, tersegmentasi berdasarkan fungsinya, dan dilindungi oleh IDS/IPS.
- Data Encryption: Enkripsi semua data sensitif di database, sehingga meskipun terjadi data breach, data tidak langsung bisa digunakan oleh pihak yang tidak berwenang.
- Monitoring dan Logging: Pantau seluruh aktivitas secara real-time, terutama untuk deteksi serangan seperti SQL Injection atau XSS yang berhasil lolos dari filter aplikasi.

Pada aplikasi web, pertahanan berlapis dapat mencakup: (1) firewall untuk menyaring akses IP, (2) otentikasi kuat untuk pengguna, (3) validasi input data pada server, dan (4) enkripsi data saat transit.

3.1.3. Prinsip 3: Fail-Safe Defaults

Prinsip fail-safe defaults menyarankan bahwa jika terjadi kegagalan dalam sistem, maka default tindakan harus berada dalam keadaan aman. Dengan kata lain, jika ada keraguan atau

kegagalan dalam proses izin akses, maka sistem harus menolak akses secara otomatis.

Penerapan dalam Sistem:

- Akses yang Diblokir Secara Default: Izin hanya diberikan jika ada otorisasi yang jelas. Sebagai contoh, semua firewall dan pengaturan akses diatur untuk menolak akses secara default.
- Otentikasi Gagal secara Aman: Jika otentikasi tidak dapat diverifikasi, pengguna tidak diberikan akses sampai identitasnya dapat dipastikan.
- Pemulihan Sistem yang Aman: Dalam situasi darurat atau kerusakan, pastikan sistem kembali ke keadaan yang aman daripada meninggalkan akses terbuka.

Keuntungan:

- Mengurangi potensi kesalahan konfigurasi yang bisa dieksploitasi.
- Mencegah akses tidak sah ketika terjadi kegagalan atau situasi ambigu.

Misalkan kita merancang aplikasi keuangan yang memiliki beberapa tingkat akses pengguna, seperti user, admin, dan super-admin. Dalam penerapan Fail-Safe Defaults, desain aplikasi akan memastikan bahwa:

- Akses yang Dibatasi Secara Default
- Kebijakan Akses Minimal
- Default Deny pada Pengaturan API
- Contingency pada Autentikasi

- Pengaturan dan Hak Akses Data
- Pengaturan Kegagalan Sistem

Misalnya, dalam mendesain aplikasi keuangan, Anda bisa menggunakan pendekatan berikut:

- Secara Default Menolak Akses: Hak akses pengguna diatur sedemikian rupa sehingga, jika ada masalah dalam proses autentikasi, akses pengguna ke sistem ditolak.
- Konfigurasi API dengan Izin Minimal: API yang digunakan hanya mengizinkan akses berdasarkan izin yang sangat spesifik, dan akses ini bisa dibatasi berdasarkan peran pengguna.
- Penanganan Kegagalan: Dalam situasi di mana aplikasi tidak dapat memverifikasi hak akses, akses akan langsung ditolak hingga sistem kembali ke kondisi normal.

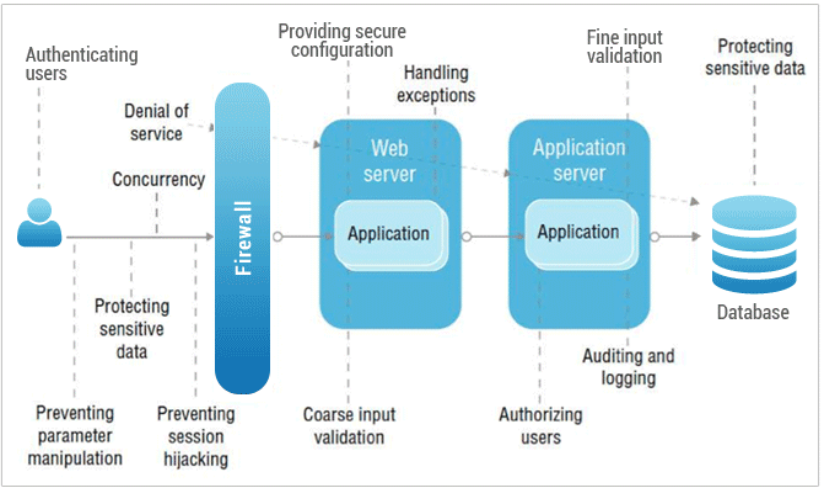
Dalam kasus kegagalan otentikasi, pengguna yang tidak dapat diverifikasi identitasnya tidak diizinkan masuk. Sistem akan meminta pengguna untuk mencoba lagi atau memverifikasi identitasnya dengan administrator.

3.2. Secure Software Architecture

Pada Secure Software Design, bagian arsitektur aplikasi yang aman berfokus pada perancangan yang mengedepankan keamanan sejak awal. Tujuannya adalah untuk meminimalkan potensi risiko dengan menggunakan prinsip-prinsip arsitektur yang solid. Dalam

konteks ini, arsitektur aplikasi yang aman memastikan bahwa setiap komponen, baik secara individu maupun keseluruhan sistem, memiliki lapisan keamanan yang memadai.

Berikut adalah elemen-elemen utama dan langkah-langkah untuk merancang arsitektur aplikasi yang aman:



Gambar 3.1 Security Architecture pada Aplikasi Web
Sumber: https://medium.com/cermati-tech/security-in-software-development-and-infrastructure-system-design-7b675c2323fc

Pola pikir keamanan dan manajemen risiko dapat diterapkan mulai dari tahap desain sistem. Menerjemahkan persyaratan — termasuk persyaratan keamanan — ke dalam desain sistem yang bisa diterapkan sebelum kita melanjutkan penerapannya adalah awal yang baik untuk pengembangan sistem yang aman.

Gambar 3.1 menunjukkan mekanisme keamanan yang bekerja ketika pengguna mengakses aplikasi berbasis web. Masalah keamanan umum pada sistem perangkat lunak atau sistem infrastruktur TI masih berkisar pada triad CIA.

3.2.1. Segregasi Komponen dan Layering (Layered Architecture)

- Pembagian Layer: Struktur aplikasi dipecah menjadi beberapa layer, seperti presentation layer, business logic layer, dan data access layer. Pembagian ini memastikan bahwa setiap lapisan memiliki peran khusus dan terbatas dalam menangani data.
- Isolasi Komponen: Setiap komponen, seperti server aplikasi, database, dan server file, dipisahkan dalam layer yang berbeda dengan batasan akses antar-layer. Ini mencegah eksploitasi dari satu layer ke layer lainnya.
- Contoh: Dalam aplikasi perbankan, presentation layer hanya mengontrol antarmuka pengguna, dan tidak memiliki akses langsung ke data access layer. Semua permintaan data pengguna harus melewati business logic layer untuk validasi dan pemrosesan.

3.2.2. Penggunaan Prinsip Least Privilege

- Setiap komponen dan modul aplikasi hanya mendapatkan hak akses minimum yang diperlukan untuk menjalankan tugasnya.
- Hak istimewa tinggi hanya diberikan kepada komponen yang benar-benar membutuhkannya, dan peran akses dikontrol dengan kebijakan role-based access control (RBAC).
- Contoh: Dalam database, hanya admin atau modul yang memiliki hak istimewa yang dapat mengakses tabel data sensitif. Pengguna aplikasi hanya mendapatkan akses ke data yang mereka perlukan untuk peran spesifik mereka.

3.2.3. Enkripsi Data dalam Transit dan Data at Rest

- Data yang dikirim antara komponen (misalnya, antara frontend dan backend atau dari aplikasi ke database) harus dienkripsi untuk mencegah eavesdropping atau manipulasi oleh pihak tidak sah.
- Enkripsi data yang disimpan (data at rest) mencegah pencurian data meskipun terjadi breach pada penyimpanan fisik atau virtual.
- Contoh: Semua informasi kartu kredit dienkripsi dalam database menggunakan AES-256 atau metode enkripsi serupa, dan komunikasi antar server dilakukan menggunakan protokol HTTPS/TLS.

3.2.4. Arsitektur Zero Trust

- Zero Trust mengedepankan prinsip "never trust, always verify," yang berarti setiap permintaan akses harus divalidasi, baik yang berasal dari luar maupun dalam sistem.
- Identitas pengguna dan perangkat diverifikasi secara kontinu menggunakan autentikasi multifaktor dan tokenisasi, memastikan bahwa setiap permintaan dianggap tidak aman hingga terbukti aman.
- Contoh: Pada aplikasi keuangan, setiap pengguna harus melewati autentikasi multifaktor setiap kali mengakses data sensitif, bahkan jika mereka sudah login. Hal ini diterapkan pada aplikasi dan API-nya.

3.2.5. Firewall Aplikasi dan Pembatasan Akses Jaringan

- Penggunaan firewall aplikasi (Web Application Firewall/WAF) untuk memfilter lalu lintas berbahaya dan mencegah serangan seperti SQL injection atau cross-site scripting (XSS).

- Penerapan kontrol akses jaringan untuk membatasi akses ke port atau IP tertentu.
- Contoh: Server aplikasi hanya membuka port yang diperlukan, seperti port 80 dan 443 untuk lalu lintas HTTP/HTTPS. Port database tidak dibuka ke internet dan hanya dapat diakses oleh server aplikasi melalui jaringan internal.

3.2.6. Audit dan Logging Terpusat

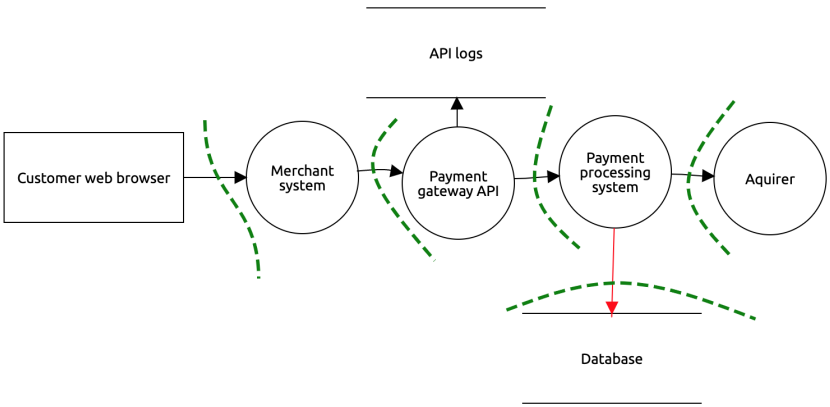
- Setiap aktivitas penting dalam sistem (misalnya, login, perubahan data, dan akses ke data sensitif) dicatat dengan baik untuk memungkinkan audit keamanan dan pelacakan.
- Logging dilakukan secara terpusat sehingga dapat dianalisis dengan mudah untuk mendeteksi pola atau serangan yang mencurigakan.
- Contoh: Setiap permintaan API atau interaksi pengguna dalam aplikasi dicatat di log terpusat dengan informasi penting seperti waktu akses, lokasi, dan tindakan yang dilakukan, untuk memungkinkan deteksi dini atas aktivitas yang mencurigakan.

3.2.7. Desain untuk Kegagalan (Fail-Safe Defaults)

- Mengatur sistem untuk default deny, yang berarti bahwa sistem akan secara otomatis menolak akses ke sumber daya atau data sensitif ketika tidak ada izin eksplisit.
- Pengaturan sistem agar secara otomatis memasuki mode aman ketika terjadi kegagalan sistem.
- Contoh: Jika terjadi kegagalan autentikasi pada API, akses ke aplikasi otomatis ditolak hingga dapat dikonfirmasi. Sistem tidak akan memberikan akses penuh karena kegagalan otentikasi.

3.3. Threat Modeling

Threat modeling adalah proses sistematis untuk mengidentifikasi, mengevaluasi, dan memprioritaskan potensi ancaman terhadap sistem atau aplikasi. Dengan menggunakan pendekatan ini, tim pengembangan dapat mengidentifikasi risiko yang paling mungkin terjadi, sehingga mereka dapat mengambil langkah untuk menguranginya.



Gambar 3.2 Contoh Threat Modeling
Sumber: https://owasp.org/www-project-security-culture/v10/6-Threat_Modelling/

Diagram aliran data model ancaman menggunakan persegi panjang untuk mewakili Aktor. Lingkaran digunakan untuk mewakili Proses, seperti aplikasi web atau API. Dan persegi panjang tanpa tepi vertikal digunakan untuk mewakili penyimpanan data, seperti database atau file konfigurasi. Panah digunakan untuk menunjukkan arus informasi antara Aktor, Proses dan penyimpanan data. Batasan kepercayaan, garis putus-putus, ditempatkan pada diagram aliran data pada arus informasi antar sistem untuk menunjukkan di mana data mengubah tingkat kepercayaannya. Terakhir, identifikasi data sensitif apa pun yang ada, baik dalam arus informasi atau penyimpanan data. Mungkin

membantu untuk berkonsultasi dengan kebijakan klasifikasi data organisasi ketika mengidentifikasi data sensitif.

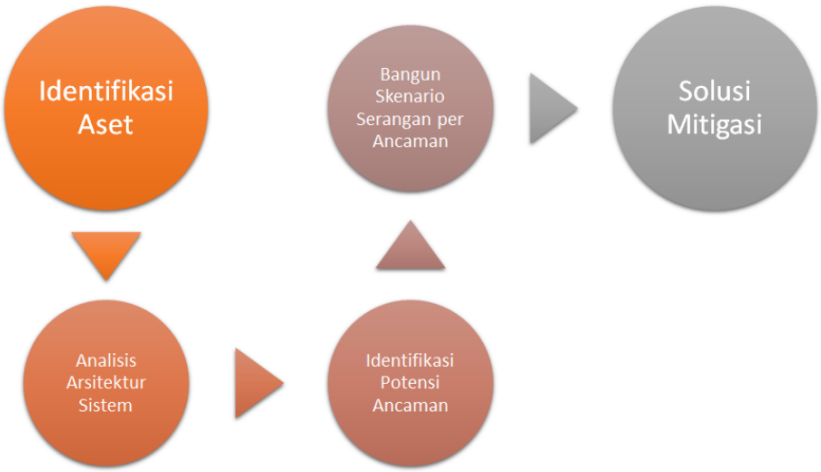
Lihat ini untuk lebih detail : https://owasp.org/www-community/Threat_Modeling_Process#sample-scope-the-work

Langkah-langkah dalam Threat Modeling melibatkan serangkaian proses yang terstruktur untuk memahami dan mengatasi potensi risiko keamanan dalam sistem teknologi informasi. Proses ini dimulai dengan:

- Identifikasi Aset Berharga (Assets): Mengidentifikasi data atau komponen yang penting, seperti data pengguna, kredensial, informasi pribadi, atau informasi keuangan, yang perlu dilindungi.
- Pemahaman tentang Arsitektur Sistem: Mendokumentasikan semua komponen aplikasi, termasuk modul, API, jaringan, dan interaksi antar-komponen.
- Identifikasi Potensi Ancaman: Menggunakan metode seperti STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) untuk mengevaluasi potensi ancaman dari berbagai perspektif.
- Membangun Skenario Serangan (Attack Scenarios): Membuat skenario serangan yang mungkin dilakukan oleh

penyerang, seperti serangan SQL injection, spoofing, atau XSS.

- Menilai dan Mengurangi Ancaman (Mitigation): Setelah ancaman diidentifikasi, solusi diterapkan untuk memitigasi atau mengurangi dampak dari ancaman tersebut.



Gambar 3.3 Langkah-Langkah Threat Modeling
(Sumber: Penulis)

Contoh Penerapan Threat Modeling di Perusahaan

- Perusahaan Keuangan (Bank Digital)
    - Deskripsi: Sebuah bank digital sedang mengembangkan aplikasi mobile banking yang memungkinkan pengguna untuk melakukan transfer dana, mengecek saldo, dan mengelola akun mereka.
- Penerapan Threat Modeling:
    - Diagram Arsitektur: Tim pengembang membuat diagram arsitektur aplikasi untuk memvisualisasikan komponen utama seperti server backend, API, dan basis data.
    - Identifikasi Ancaman: Menggunakan metode STRIDE, tim mengidentifikasi potensi ancaman:
        - Spoofing: Pengguna palsu mencoba mengakses akun dengan memalsukan identitas.
        - Tampering: Serangan untuk mengubah data transaksi saat dilakukan.
        - Information Disclosure: Data sensitif pengguna mungkin diungkap saat komunikasi tidak terenkripsi.
        - Denial of Service: Upaya untuk menjatuhkan layanan melalui serangan berbobot tinggi.
        - Elevation of Privilege: Pengguna biasa memperoleh hak akses administrator.

- Mitigasi: Mengembangkan langkah-langkah untuk mitigasi, termasuk menerapkan otentikasi dua faktor, enkripsi data dalam transit, dan pengujian keamanan berkala.

Contoh Penerapan Threat Modeling di Perusahaan

- Perusahaan Teknologi (Software as a Service/SaaS)
    - Deskripsi: Sebuah perusahaan SaaS yang menyediakan platform manajemen proyek ingin memastikan aplikasi mereka aman dari berbagai ancaman.
- Penerapan Threat Modeling:
    - Pembuatan Model: Tim memetakan semua entitas dalam aplikasi, termasuk pengguna, database, dan integrasi pihak ketiga.
    - Analisis Ancaman: Melakukan diskusi dengan tim pengembangan dan keamanan untuk mengidentifikasi ancaman yang mungkin, misal:
        - Insider Threats: Karyawan dengan akses dapat mengekstrak data yang sensitif.
        - Vulnerabilities in Third-Party Libraries: Menggunakan pustaka kode pihak ketiga yang memiliki kerentanan.
    - Evaluasi: Menggunakan pendekatan DREAD (Damage, Reproducibility, Exploitability, Affected Users, Discoverability) untuk menilai dampak dan kemungkinan setiap ancaman.

- Dokumentasi dan Tindak Lanjut: Membuat dokumentasi terperinci dari semua temuan dan rencana mitigasi yang akan diintegrasikan ke dalam siklus pengembangan.

CHAPTER 4
SECURE SOFTWARE IMPLEMENTATION

Materi:

- Teknik input validation, output encoding, dan error handling.
- Menggunakan library atau framework yang aman.
- Panduan secure coding (misalnya, OWASP, SANS)

Durasi:

- 2 Jam

Tujuan:

- Menerapkan praktik secure coding untuk mencegah vulnerability.

Target Pembaca:

- Karyawan IT (SysAdmin, Network Engineer, DevOps)

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

4.1. Input Validation

Input validation adalah proses memeriksa dan memverifikasi bahwa data yang dimasukkan ke dalam aplikasi adalah data yang valid dan aman. Teknik ini mencegah berbagai serangan yang memanfaatkan input yang tidak diharapkan, seperti SQL Injection, Cross-Site Scripting (XSS), dan Buffer Overflow.

Jenis-Jenis Input Validation:

- Client-Side Validation: Dilakukan di sisi klien untuk kenyamanan pengguna (misalnya, pemeriksaan format email sebelum mengirim form), tetapi tidak cukup untuk keamanan karena dapat dilewati.
- Server-Side Validation: Dilakukan di server dan merupakan langkah yang lebih aman untuk memastikan bahwa data yang diterima adalah benar-benar valid.

Contoh Praktik Input Validation:

- Jenis Data: Pastikan input sesuai dengan jenis data yang diharapkan, seperti hanya menerima angka pada kolom usia.
- Format dan Panjang: Batasi panjang input dan format, seperti panjang password minimum 8 karakter.
- Range Check: Batasi rentang angka atau nilai, misalnya umur harus antara 18 dan 99.
- White-listing: Terima hanya karakter atau input yang valid, seperti hanya menerima karakter A-Z untuk nama.

Sekarang kita akan mencoba bagaimana perbandingan antara kode tanpa input validasi dengan yang menggunakan input validasi dalam contoh berikut:

```php
<?php
// Mendapatkan input pengguna
$username = $_POST['username'];
$password = $_POST['password'];

// Query tanpa input validation
$query = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
$result = mysqli_query($conn, $query);

if (mysqli_num_rows($result) > 0) {
    echo "Login berhasil!";
} else {
    echo "Username atau password salah.";
}
?>
```

Gambar 4.1 Contoh Kode Tanpa Input Validasi
(Sumber: Penulis)

Di sini, input pengguna disisipkan langsung ke dalam query SQL tanpa adanya perlindungan apa pun. Akibatnya, aplikasi ini rentan terhadap serangan SQL Injection.

Bagaimana SQL Injection Bisa Terjadi?

Jika pengguna mencoba memasukkan input sebagai berikut:

- Username: ' OR '1'='1
- Password: anything

Maka, query SQL akan menjadi:

```sql
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = 'anything'
```

Bagian OR '1'='1' akan selalu bernilai benar, sehingga query ini akan mengembalikan semua baris dalam tabel users. Hasilnya, login akan berhasil terlepas dari data pengguna yang sebenarnya karena SQL Injection berhasil mengeksploitasi query tanpa validasi.

Untuk mencegah SQL Injection, kita bisa menggunakan prepared statements, yang akan memastikan bahwa input dari pengguna tidak dapat mengubah struktur query SQL.

```php
<?php
// Mendapatkan input pengguna
$username = $_POST['username'];
$password = $_POST['password'];

// Menggunakan prepared statements
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
$stmt->bind_param("ss", $username, $password);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    echo "Login berhasil!";
} else {
    echo "Username atau password salah.";
}
?>
```

Gambar 4.3 Contoh Kode dengan Input Validasi
(Sumber: Penulis)

Di sini, kita menggunakan ? sebagai placeholder untuk username dan password, yang kemudian diikat dengan metode bind_param. Dengan cara ini, nilai yang dimasukkan oleh pengguna tidak akan diinterpretasikan sebagai bagian dari query SQL, sehingga mencegah SQL Injection.

Jadi, pertanyaannya adalah Bagaimana Input Validasi Mencegah SQL Injection?

- Prepared Statements: Dengan menggunakan prepared statements, input pengguna tidak disisipkan langsung ke dalam query SQL. Database akan memperlakukan input sebagai data murni, bukan sebagai bagian dari instruksi SQL, sehingga karakter yang biasanya memicu SQL

Injection (seperti ', --, atau OR) tidak akan berfungsi sebagai perintah SQL.

- Bind Parameters: Dalam prepared statements, bind parameters memungkinkan untuk mengikat data tanpa mengubah struktur query. Ini memastikan bahwa semua input diperlakukan hanya sebagai data, bukan kode.

4.2. Output Encoding

Output encoding adalah proses mengubah data menjadi format yang aman sebelum ditampilkan kepada pengguna. Ini penting untuk mencegah data yang tidak aman dieksekusi sebagai kode oleh browser atau aplikasi lain, yang dapat menyebabkan serangan XSS (Cross-Site Scripting). Jenis-Jenis Output Encoding:

- HTML Encoding: Mengubah karakter khusus menjadi entities HTML untuk mencegah eksekusi kode dalam HTML.
- URL Encoding: Mengonversi karakter yang tidak aman menjadi format yang aman untuk URL.
- JavaScript Encoding: Mengubah karakter khusus agar tidak dieksekusi sebagai kode JavaScript di dalam halaman web.

Contoh Praktik Output Encoding: Misalnya, untuk memastikan karakter khusus di-encode sebelum ditampilkan di HTML:

```
<!-- PHP HTML encoding untuk mencegah XSS -->
<?php echo htmlspecialchars($userInput, ENT_QUOTES, 'UTF-8'); ?>
```

Gambar 4.3. Contoh Encode pada HTML
(Sumber: Penulis)

Contoh di atas mengubah tanda kutip, simbol kurang dari (<), dan lainnya menjadi entitas HTML, sehingga tidak dijalankan sebagai

kode. Sekarang kita akan mencoba bagaimana perbandingan antara kode tanpa output encoding dengan yang menggunakan output encoding dalam contoh berikut:

```
from flask import Flask, request, render_template_string

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    comment = ""
    if request.method == "POST":
        comment = request.form.get("comment")  # Tidak ada encoding output
    # Render komentar langsung di HTML
    html = f"""
    <html>
        <body>
            <h1>Komentar Pengguna</h1>
            <form method="POST">
                <textarea name="comment"></textarea>
                <button type="submit">Submit</button>
            </form>
            <p>{comment}</p> <!-- Menampilkan komentar tanpa encoding -->
        </body>
    </html>
    """
    return render_template_string(html)

if __name__ == "__main__":
    app.run(debug=True)
```

Gambar 4.4. Contoh Kode Tanpa Output Encoding
(Sumber: Penulis)

Di sini, kita mengambil komentar pengguna dan langsung menampilkannya di halaman HTML tanpa output encoding. Karena itu, jika pengguna memasukkan skrip seperti berikut:

```
<script>alert('XSS Attack!');</script>
```

Ketika halaman dimuat, JavaScript ini akan dijalankan, menampilkan pop-up "XSS Attack!". Ini adalah serangan XSS yang bisa berbahaya, terutama jika digunakan untuk mencuri cookie pengguna atau mengubah halaman.

Untuk mencegah serawngan XSS, maka kode sebelumnya perlu ditambah fungsi output encoding. Berikut contoh kode yang sudah menerapkan output encoding yang dapat dilihat pada gambar 4.5.

Pada contoh ini, kita menggunakan escape(comment) untuk melakukan output encoding pada komentar pengguna. Fungsi escape akan mengonversi karakter khusus seperti <, >, &, dan " ke dalam bentuk HTML yang aman, misalnya:

- < menjadi <
- > menjadi >
- & menjadi &
- " menjadi "

Jika pengguna mencoba memasukkan "<script>alert('XSS Attack!');</script>", teks tersebut akan diubah menjadi <script>alert('XSS Attack!');</script> dan ditampilkan di halaman sebagai teks biasa, bukan kode yang dieksekusi.

```python
from flask import Flask, request, render_template_string
from html import escape  # Import fungsi untuk encoding output

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    comment = ""
    if request.method == "POST":
        comment = request.form.get("comment")
        comment = escape(comment)  # Encoding output untuk mencegah XSS
    # Render komentar dengan encoding
    html = f"""
    <html>
        <body>
            <h1>Komentar Pengguna</h1>
            <form method="POST">
                <textarea name="comment"></textarea>
                <button type="submit">Submit</button>
            </form>
            <p>{comment}</p> <!-- Komentar aman dengan encoding -->
        </body>
    </html>
    """
    return render_template_string(html)

if __name__ == "__main__":
    app.run(debug=True)
```

Gambar 4.5 Contoh kode dengan output encoding
(Sumber: Penulis)

4.3. Error Handling

Error handling adalah proses menangani dan menampilkan pesan kesalahan dengan aman. Pesan kesalahan harus memberi informasi yang cukup kepada pengguna atau admin tentang adanya masalah, namun tidak memberikan informasi berlebihan

yang dapat dimanfaatkan oleh penyerang untuk mengenali kelemahan sistem.

Prinsip-Prinsip Error Handling:

- Minimal Disclosure: Batasi informasi yang ditampilkan kepada pengguna. Hindari menampilkan pesan kesalahan teknis (misalnya, "Syntax Error in Line 10") karena ini dapat membantu penyerang.
- Logging Kesalahan: Simpan detail kesalahan pada log server yang hanya dapat diakses oleh tim pengembang atau admin.
- Generik untuk Pengguna, Detail untuk Admin: Berikan pesan yang ramah pengguna seperti "Terjadi kesalahan, silakan coba lagi," dan simpan informasi teknis di log server.

Contoh Praktik Error Handling: Misalnya, menampilkan pesan kesalahan tanpa mengungkapkan rincian yang spesifik:

```php
try {
    // Kode eksekusi
} catch (Exception $e) {
    // Pesan kesalahan untuk pengguna
    echo "Terjadi kesalahan, silakan coba lagi nanti.";

    // Log detail kesalahan untuk admin
    error_log($e->getMessage());
}
```

Gambar 4.6 Contoh Error Handling
(Sumber: Penulis)

Untuk penerapan error handling ini, kita bisa membuat contoh kode yang sudah menerapkan error handling supaya pembaca dapat memahami perbedaannya seperti apa. Berikut contoh kode yang menggunakan error handling dengan tepat:

```python
from flask import Flask, request
import logging

app = Flask(__name__)

# Konfigurasi logging
logging.basicConfig(filename="app.log", level=logging.ERROR)

@app.route("/login", methods=["POST"])
def login():
    username = request.form.get("username")
    password = request.form.get("password")
    # Koneksi ke database
    try:
        user = database.get_user(username, password)
        if user:
            return "Login berhasil!"
        else:
            return "Login gagal!"
    except Exception as e:
        # Log error tanpa menampilkannya ke pengguna
        logging.error("Error pada login: %s", e)
        return "Terjadi kesalahan, silakan coba lagi nanti."

if __name__ == "__main__":
    app.run(debug=False)  # Nonaktifkan debug mode di production
```

Gambar 4.7 Contoh kode dengan error handling
(Sumber: Penulis)

Berikut penjelasan untuk poin-poin penting dalam kode diatas:

- Logging Error: Daripada menampilkan error langsung, error dicatat dalam file log (app.log) yang hanya bisa diakses oleh pengembang atau admin sistem.
- Pesan Umum untuk Pengguna: Pengguna hanya melihat pesan umum seperti "Terjadi kesalahan, silakan coba lagi nanti," sehingga tidak ada informasi teknis yang bocor.
- Nonaktifkan Debug Mode: Di lingkungan production, debug mode dinonaktifkan agar error tidak ditampilkan secara eksplisit.

4.4. Library atau Framework yang Aman

Library dan framework aman menyediakan fungsionalitas dan pendekatan standar yang sudah diuji untuk mengatasi kerentanan. Keunggulan menggunakan library atau framework yang aman antara lain:

- Penghematan Waktu dan Usaha: Alih-alih mengembangkan fitur keamanan dari awal, developer bisa langsung memanfaatkan fungsionalitas yang ada di library atau framework yang terpercaya.
- Pemutakhiran Keamanan: Library atau framework yang aktif dikelola akan sering menerima pembaruan keamanan yang menutup celah kerentanan.
- Kepatuhan terhadap Standar Keamanan: Banyak library dan framework didesain sesuai dengan standar keamanan seperti OWASP, PCI-DSS, atau NIST, yang membantu developer mematuhi regulasi keamanan.
- Meminimalkan Kesalahan Manual: Library atau framework yang aman menyediakan metode yang telah

teruji untuk validasi input, autentikasi, enkripsi, dan sebagainya, mengurangi risiko human error.

Contoh Fitur Keamanan di Library dan Framework sangat beragam. Berikut adalah beberapa contoh fitur keamanan di library atau framework populer:

- Validasi Input dan Output Encoding: Misalnya, Express.js untuk Node.js menyediakan fitur middleware yang membantu memvalidasi dan memfilter input.
- Enkripsi dan Dekripsi Data: Libraries seperti bcrypt untuk hashing kata sandi atau CryptoJS untuk enkripsi data memudahkan developer menerapkan kriptografi.
- Autentikasi dan Otentikasi SSO: Framework seperti Django (Python) dan Spring Security (Java) memiliki sistem autentikasi bawaan dan mendukung otentikasi berbasis token atau SSO, sehingga memudahkan developer membangun autentikasi yang aman.

Pada contoh di bawah ini, kata sandi disimpan dalam plaintext atau bentuk asli di database, yang rentan jika database terekspos.

```
# Tidak menggunakan library aman untuk hashing
def login(username, password):
    # Verifikasi kata sandi (plaintext) secara langsung
    stored_password = get_password_from_database(username)
    if password == stored_password:
        print("Login berhasil")
    else:
        print("Login gagal")
```

Gambar 4.8 Contoh kode tanpa bcrypt
(Sumber: Penulis)

Berikut ini contoh ketika kode sudah menggunakan secure library dengan bcrypt:

```python
import bcrypt

# Fungsi untuk menyimpan password dengan hashing
def store_password(username, password):
    hashed = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
    save_password_to_database(username, hashed)

# Fungsi untuk autentikasi
def login(username, password):
    stored_password = get_password_from_database(username)
    # Verifikasi kata sandi dengan bcrypt
    if bcrypt.checkpw(password.encode('utf-8'), stored_password):
        print("Login berhasil")
    else:
        print("Login gagal")
```

Gambar 4.9 Contoh kode dengan bcrypt
(Sumber: Penulis)

Pada contoh di atas:

- bcrypt digunakan untuk meng-hash kata sandi sebelum disimpan ke database.
- Ketika user login, input kata sandi di-hash terlebih dahulu dan dibandingkan dengan hash yang tersimpan.

Dengan ini, jika database bocor, penyerang tidak akan dapat langsung melihat kata sandi pengguna.

Adapun secure library yang dapat kita gunakan adalah Express.js. Library Express.js adalah salah satu framework yang populer di kalangan pengembang Node.js untuk membangun aplikasi web. Express.js menawarkan berbagai fitur yang membantu dalam pengembangan aplikasi dengan cara yang lebih efisien dan terstruktur. Berikut ini contoh kode tanpa express.js:

```
const express = require('express');
const app = express();
app.use(express.json());

const users = [
    { username: 'user1', password: 'password123' } // Password disimpan sebagai plaintext
];

// Fungsi login tanpa enkripsi atau hashing
app.post('/login', (req, res) => {
    const { username, password } = req.body;
    const user = users.find(user => user.username === username);

    if (user && user.password === password) {
        res.status(200).send("Login berhasil!");
    } else {
        res.status(401).send("Username atau password salah.");
    }
});

app.listen(3000, () => {
    console.log("Server berjalan di port 3000");
});
```

Gambar 4.10 Contoh kode tanpa express.js
(Sumber: Penulis)

Pada kode diatas :

- Kata sandi disimpan sebagai plaintext.
- Rentan terhadap kebocoran data, karena jika data terekspos, kata sandi dapat langsung dilihat oleh penyerang.
- Tidak ada pembatasan atau validasi untuk serangan brute-force atau rate limiting.

Untuk menangani masalah diatas, kita bisa menggunakan library express.js untuk mengatasi masalah kebocoran data. Berikut contoh kode dengan menggunakan express.js:

const express = require('express');

```
const bcrypt = require('bcrypt');

const helmet = require('helmet');

const app = express();

app.use(express.json());

app.use(helmet()); // Menambahkan header keamanan bawaan dengan Helmet

const users = [];

// Fungsi untuk registrasi pengguna baru dengan hashing password

app.post('/register', async (req, res) => {

const { username, password } = req.body;

// Meng-hash password sebelum menyimpannya

const hashedPassword = await bcrypt.hash(password, 10);

users.push({ username, password: hashedPassword });

res.status(201).send("Pengguna berhasil didaftarkan!");

});

// Fungsi login dengan validasi menggunakan bcrypt

app.post('/login', async (req, res) => {

const { username, password } = req.body;

const user = users.find(user => user.username === username);

if (user) {

// Membandingkan password yang di-hash dengan password yang dimasukkan
```

```
const match = await bcrypt.compare(password, user.password);
if (match) {
res.status(200).send("Login berhasil!");
} else {
res.status(401).send("Username atau password salah.");
}
} else {
res.status(401).send("Username atau password salah.");
}
});
app.listen(3000, () => {
console.log("Server berjalan di port 3000");
});
```

Kode diatas merupakan contoh dasar penerapan secure coding dalam autentikasi menggunakan library yang mendukung keamanan, dan bisa menjadi titik awal untuk lebih banyak peningkatan keamanan dalam aplikasi Express.js.

Jika menggunakan laravel, mereka juga menyediakan secure library yang bisa kita gunakan untuk mengamankan kode. Misalkan ada Form Input Komentar (contoh pada resources/views/comment.blade.php) tanpa menggunakan library:

```
<form action="{{ route('comment.store') }}" method="POST">
    @csrf
    <textarea name="comment" placeholder="Masukkan komentar Anda"></textarea>
    <button type="submit">Submit</button>
</form>
```

Kemudian ada fungsi Menyimpan Komentar ke Database:

```
// Controller: CommentController.php
public function store(Request $request)
{
    $comment = new Comment();
    $comment->text = $request->comment; // Tidak ada sanitasi input
    $comment->save();

    return redirect()->route('comment.index');
}
```

Setelah itu kita akan Menampilkan Komentar di Halaman Tanpa XSS Protection:

```
<!-- Di file comment.blade.php -->
@foreach($comments as $comment)
    <p>{!! $comment->text !!}</p> <!-- Tidak menggunakan blade escaping -->
@endforeach
```

Dengan penggunaan {!! !!}, input akan ditampilkan tanpa di-escape. Jika pengguna memasukkan sesuatu seperti <script>alert('XSS!')</script>, skrip tersebut akan dieksekusi, yang merupakan kerentanan XSS.

Untuk menangani kasus ini di laravel, maka kita perlu menggunakan secure library yang disediakan laravel sebagai berikut.

Untuk Form Input Komentar, tidak ada perubahan di sini. Dan untuk Menyimpan Komentar ke Database, juga tidak ada perubahan. Tapi, kita akan fokus pada fungsi untuk menampilkan komentar. Disini kita akan Menampilkan Komentar dengan Blade Escaping untuk Melindungi dari XSS:

```
<!-- Di file comment.blade.php -->
@foreach($comments as $comment)
    <p>{{ $comment->text }}</p> <!-- Menggunakan blade escaping untuk menghindari XSS -->
@endforeach
```

- Menggunakan {{ $comment->text }} di blade view secara otomatis meng-escape karakter berbahaya, mencegah skrip seperti <script> dijalankan.
- Jika pengguna mencoba memasukkan <script>alert('XSS!')</script>, Laravel akan menampilkan teks sebagai <script>alert('XSS!')</script>, sehingga tidak dieksekusi.

4.5. Panduan Secure Coding

Dalam dunia pengembangan perangkat lunak yang semakin kompleks, keamanan menjadi salah satu aspek terpenting yang tidak boleh diabaikan. Serangan siber yang semakin canggih dan meningkatnya jumlah pelanggaran data menunjukkan betapa pentingnya menanamkan prinsip-prinsip keamanan ke dalam kode yang ditulis. Inilah alasan mengapa panduan secure coding (pengkodean aman) menjadi sangat vital bagi setiap pengembang. Panduan ini bertujuan untuk memberikan pedoman yang jelas dan praktik terbaik dalam menulis kode yang tidak hanya berfungsi dengan baik tetapi juga aman dari potensi ancaman. Disini kita akan membahas 2 (dua) standar secure coding yang umum dan populer digunakan.

4.5.1. OWASP (Open Web Application Security Project)

OWASP adalah organisasi nirlaba yang menyediakan berbagai sumber daya untuk keamanan aplikasi. Salah satu dokumen paling terkenal adalah OWASP Top 10, yang mengidentifikasi 10 ancaman keamanan aplikasi web paling kritis.

OWASP Top 10 diperbarui secara berkala untuk mencerminkan tren ancaman terbaru. Beberapa ancaman yang sering dibahas dalam panduan OWASP antara lain:

- Injection: Mencegah SQL, OS, dan LDAP Injection dengan validasi input dan parameterized queries.
- Broken Authentication: Memastikan autentikasi yang kuat dan menghindari kebocoran data pengguna.
- Sensitive Data Exposure: Melindungi data dengan enkripsi dan memastikan kebijakan perlindungan data yang kuat.
- Security Misconfiguration: Mengamankan konfigurasi server dan database.
- Cross-Site Scripting (XSS): Menerapkan output encoding untuk mencegah skrip yang disisipkan oleh pengguna.
- Insecure Deserialization: Mencegah deserialization yang berpotensi merusak integritas data.

OWASP juga menyediakan berbagai proyek terkait panduan secure coding, di antaranya:

- OWASP Application Security Verification Standard (ASVS): Standar untuk memverifikasi tingkat keamanan aplikasi.
- OWASP Secure Coding Practices Checklist: Daftar periksa langkah-langkah praktis untuk mengamankan kode pada setiap tahap pengembangan.
- OWASP Proactive Controls: Daftar 10 langkah proaktif untuk membantu pengembang merancang aplikasi yang aman.

Dengan menerapkan panduan OWASP, pengembang dapat memiliki pemahaman tentang ancaman utama yang dapat dieksploitasi dan menerapkan praktik terbaik untuk mencegahnya sejak awal.

OWASP Application Security Verification Standard (ASVS)

ASVS menyediakan panduan terperinci untuk pengembang dan auditor dalam menilai keamanan aplikasi. Dengan mengikuti standar ini, organisasi dapat memastikan bahwa aplikasi mereka dibangun dengan perlindungan yang tepat terhadap berbagai ancaman keamanan. Standar ini mencakup berbagai aspek, mulai dari autentikasi dan manajemen sesi hingga perlindungan data dan keamanan komunikasi. Dengan menerapkan ASVS, tim pengembang dapat meningkatkan kualitas dan kepercayaan pengguna terhadap aplikasi mereka, serta meminimalkan risiko kebocoran data atau serangan siber lainnya.

Contoh Checklist untuk Input Validasi dapat dilihat pada gambar 4.11.

#	Description	L1	L2	L3
5.1.1	Verify that the application has defenses against HTTP parameter pollution attacks, particularly if the application framework makes no distinction about the source of request parameters (GET, POST, cookies, headers, or environment variables).	✓	✓	✓
5.1.2	Verify that frameworks protect against mass parameter assignment attacks, or that the application has countermeasures to protect against unsafe parameter assignment, such as marking fields private or similar. (C5)	✓	✓	✓
5.1.3	Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers, cookies, batch files, RSS feeds, etc) is validated using positive validation (allow lists). (C5)	✓	✓	✓
5.1.4	Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers, e-mail addresses, telephone numbers, or validating that two related fields are reasonable, such as checking that suburb and zip/postcode match). (C5)	✓	✓	✓
5.1.5	Verify that URL redirects and forwards only allow destinations which appear on an allow list, or show a warning when redirecting to potentially untrusted content.	✓	✓	✓

Gambar 4.11 Checklist Input Validasi dari OWASP ASVS
(Sumber: OWASP)

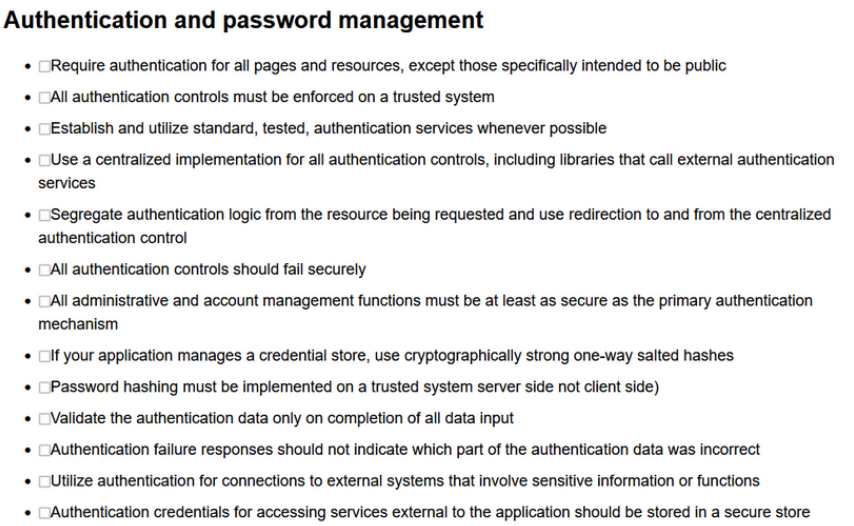
dan untuk Untuk dokumen OWASP ASVS dapat di download pada link berikut :

https://raw.githubusercontent.com/OWASP/ASVS/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

OWASP Secure Coding Practices Checklist

OWASP Secure Coding Practices Checklist adalah panduan yang lebih ringkas dan praktis, dirancang untuk memberikan langkah-langkah langsung yang dapat diterapkan oleh pengembang selama proses pengkodean. Checklist ini fokus pada penerapan praktik pengkodean yang aman, dengan tujuan mengurangi kerentanan keamanan pada aplikasi sejak tahap awal pengembangan.

Checklist ini juga dilengkapi dengan tips praktis dan contoh implementasi yang dapat membantu pengembang memahami dan menerapkan prinsip-prinsip keamanan dengan lebih efektif.



Authentication and password management

- ☐ Require authentication for all pages and resources, except those specifically intended to be public
- ☐ All authentication controls must be enforced on a trusted system
- ☐ Establish and utilize standard, tested, authentication services whenever possible
- ☐ Use a centralized implementation for all authentication controls, including libraries that call external authentication services
- ☐ Segregate authentication logic from the resource being requested and use redirection to and from the centralized authentication control
- ☐ All authentication controls should fail securely
- ☐ All administrative and account management functions must be at least as secure as the primary authentication mechanism
- ☐ If your application manages a credential store, use cryptographically strong one-way salted hashes
- ☐ Password hashing must be implemented on a trusted system server side not client side)
- ☐ Validate the authentication data only on completion of all data input
- ☐ Authentication failure responses should not indicate which part of the authentication data was incorrect
- ☐ Utilize authentication for connections to external systems that involve sensitive information or functions
- ☐ Authentication credentials for accessing services external to the application should be stored in a secure store

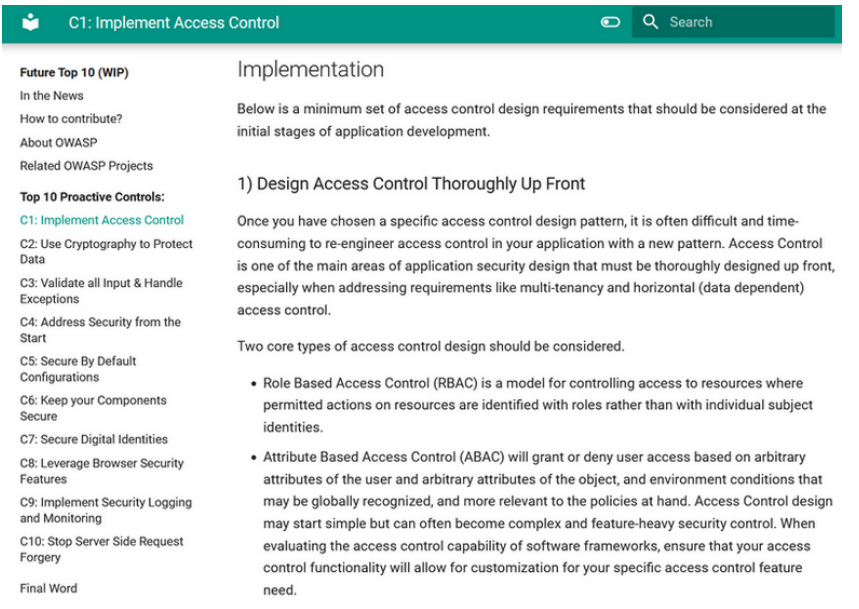
Gambar 4.12 Checklist Authentication and Password Management dari OWASP SCPC
(Sumber: OWASP)

Selain itu, OWASP Secure Coding Practices Checklist dapat digunakan sebagai alat pelatihan untuk meningkatkan kesadaran dan keterampilan tim pengembang dalam menghadapi tantangan keamanan aplikasi modern. Untuk dokumen OWASP Secure COding Practices Checklist dapat di download pada link berikut :

https://raw.githubusercontent.com/OWASP/ASVS/v4.0.3/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.3-en.pdf

OWASP Proactive Controls

OWASP Proactive Controls merupakan satu set panduan yang direka untuk membantu pembangun perisian meningkatkan keselamatan aplikasi mereka. Ia menyediakan sepuluh amalan terbaik yang setiap pembangun perlu pertimbangkan semasa fasa pembangunan perisian. Antara langkah penting termasuklah penggunaan kawalan akses yang ketat, perlindungan data sensitif, dan pengecualian suntikan kod yang berbahaya. Dengan mengamalkan kawalan ini, organisasi dapat mengurangkan risiko keselamatan dan melindungi aplikasi mereka daripada ancaman siber yang berpotensi. Panduan ini amat berguna dalam mencipta aplikasi yang kukuh dan boleh dipercayai, memastikan keselamatan pengguna dan data mereka terjamin.



Gambar 4.13 Implement Access Control pada OWASP PC
(Sumber: OWASP)

Untuk dokumen OWASP Proactive COntrol dapat di akses pada link berikut :

https://top10proactive.owasp.org/the-top-10/

4.5.2. SANS (SysAdmin, Audit, Network, and Security Institute)

SANS adalah organisasi yang menyediakan pelatihan keamanan dan panduan praktik terbaik, termasuk dalam secure coding. Salah satu proyek terpopuler adalah SANS Top 25 Most Dangerous Software Errors, yang bekerja sama dengan CWE (Common Weakness Enumeration). SANS Top 25 merinci kelemahan keamanan yang sering muncul dalam pengembangan perangkat lunak. Contoh dari daftar tersebut meliputi:

- Improper Neutralization of Input During Web Page Generation (Cross-Site Scripting): Menyediakan panduan untuk mencegah XSS.
- Improper Access Control (Authorization): Langkah-langkah untuk memastikan akses terbatas ke data sensitif.
- Uncontrolled Resource Consumption (Denial-of-Service): Mencegah eksploitasi yang bisa mengakibatkan crash.
- Buffer Overflow: Panduan untuk menghindari buffer overflow dengan pembatasan memori.

Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2022
1	CWE-787	Out-of-bounds Write	63.72	70	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.54	4	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	34.27	6	0
4	CWE-416	Use After Free	16.71	44	+3
5	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	15.65	23	+1
6	CWE-20	Improper Input Validation	15.50	35	-2
7	CWE-125	Out-of-bounds Read	14.60	2	-2
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.11	16	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.73	0	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	10.41	5	0
11	CWE-862	Missing Authorization	6.90	0	+5
12	CWE-476	NULL Pointer Dereference	6.59	0	-1
13	CWE-287	Improper Authentication	6.39	10	+1
14	CWE-190	Integer Overflow or Wraparound	5.89	4	-1
15	CWE-502	Deserialization of Untrusted Data	5.56	14	-3
16	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	4.95	4	+1
17	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.75	7	+2
18	CWE-798	Use of Hard-coded Credentials	4.57	2	-3
19	CWE-918	Server-Side Request Forgery (SSRF)	4.56	16	+2
20	CWE-306	Missing Authentication for Critical Function	3.78	8	-2
21	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.53	8	+1
22	CWE-269	Improper Privilege Management	3.31	5	+7
23	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.30	6	+2
24	CWE-863	Incorrect Authorization	3.16	0	+4
25	CWE-276	Incorrect Default Permissions	3.16	0	-5

Gambar 4.14 Contoh SANS Top 25 Vulnerabilities
(Sumber: SANS)

4.6. Studi Kasus

Let's Play Secure Coding Dojo

Kunjungi dan daftar di website berikut :

https://securecodingdojo.owasp.org/public/index.html

Kemudian ikuti instruksi dan jawab pertanyaan yang ada di website tersebut!

Gambar 4.15 Halaman Website Secure Coding Dojo
(Sumber: OWASP)

SANS juga memiliki panduan khusus yang dapat diterapkan langsung oleh pengembang, seperti:

- SANS Software Security Institute: Rangkaian pelatihan yang komprehensif untuk menanamkan kebiasaan secure coding.
- SANS Coding Dojo: Program simulasi pelatihan untuk mengidentifikasi dan memperbaiki celah keamanan dalam aplikasi.

CHAPTER 5
SECURE SOFTWARE TESTING

Materi:

- Jenis-jenis pengujian keamanan (static, dynamic, penetration testing).
- Tools untuk testing (misalnya, OWASP ZAP, Burp Suite).
- Metode fuzz testing dan input validation testing.

Durasi:

- 2 Jam

Tujuan:

- Menerapkan dan Menguji keamanan aplikasi dengan teknik pengujian statis & dinamis, serta penetration testing.

Target Pembaca:

- Karyawan IT (SysAdmin, Network Engineer, DevOps)

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

5.1. Software Testing

Dalam konteks secure software testing, ada tiga jenis pengujian keamanan utama yang dapat digunakan untuk memastikan aplikasi aman dari kerentanan yang berpotensi dieksploitasi oleh penyerang. Berikut adalah pembahasan tentang static testing, dynamic testing, dan penetration testing.

5.1.1. Static Application Security Testing – SAST

Static Testing adalah metode pengujian keamanan yang dilakukan dengan menganalisis kode sumber aplikasi tanpa menjalankan aplikasi tersebut. Tes ini dilakukan di tahap awal pengembangan (pre-execution) untuk mendeteksi kerentanan keamanan secara langsung dalam kode.

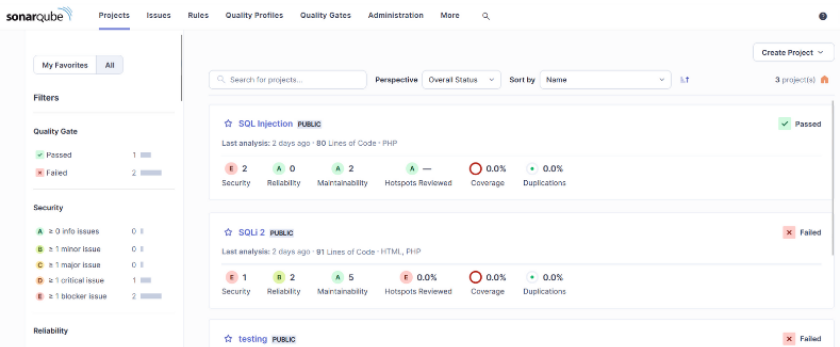
Berikut adalah langkah-langkah umum dalam menggunakan SAST sebagai bagian dari praktik secure coding:

- Pemilihan Alat SAST yang Sesuai
    - Alat SAST seperti SonarQube, Checkmarx, Fortify, dan Veracode memiliki kemampuan untuk menganalisis bahasa pemrograman tertentu dan standar keamanan.
- Analisis Kode Secara Berkala
    - Lakukan pemindaian kode secara rutin, terutama setelah setiap commit atau perubahan signifikan. Ini memungkinkan tim mengidentifikasi masalah keamanan sejak awal dan memperbaikinya segera.
- Review Hasil SAST dan Prioritaskan Perbaikan
    - SAST menghasilkan laporan dengan daftar kerentanan yang ditemukan, lengkap dengan tingkat keparahan dan rekomendasi perbaikan.

- Tindak Lanjut dengan Peningkatan Kode Aman
    - Berdasarkan hasil dari SAST, pengembang dapat melakukan perubahan pada kode agar sesuai dengan praktik secure coding. Misalnya, menambahkan validasi input, menulis ulang query untuk menghindari SQL Injection, atau memperbaiki sanitasi data untuk mencegah XSS.

Contoh Aplikasi SAST - SonarQube

SonarQube adalah alat analisis kode statis yang memeriksa kualitas kode dalam proyek perangkat lunak. Alat ini membantu pengembang mengidentifikasi bug, kerentanan keamanan, dan masalah pemeliharaan. Mendukung berbagai bahasa pemrograman, SonarQube dapat diintegrasikan dengan sistem kontrol versi dan alat CI/CD. Selain itu, SonarQube memberikan laporan komprehensif tentang kesehatan kode, mendukung peningkatan kualitas dan keandalan perangkat lunak secara berkelanjutan.



Gambar 5.1 Tampilan Aplikasi SonarQube
(Sumber: https://www.sonarsource.com)

Tools SonarQube tersedia untuk Community yang gratis (Free). Untuk mendapatkan tools tsb, dapat klik pada link berikut : https://www.sonarsource.com/

5.1.2. Dynamic Application Security Testing – DAST

Dynamic Application Security Testing (DAST) adalah metode pengujian keamanan yang berfokus pada pengujian aplikasi saat aplikasi berjalan, tanpa melihat kode sumbernya. DAST mengevaluasi aplikasi dari sudut pandang penyerang eksternal dengan mencoba berbagai teknik serangan untuk menemukan potensi kerentanan. Dalam konteks secure coding, DAST membantu mengidentifikasi masalah keamanan yang mungkin terlewatkan atau tidak tampak pada analisis statis (SAST) dan memeriksa apakah implementasi dari kode sudah aman dalam lingkungan nyata.

DAST adalah metode black-box testing yang dilakukan dari perspektif pengguna akhir atau penyerang potensial. Pengujian ini tidak membutuhkan akses ke kode sumber, melainkan bekerja dengan mengirimkan permintaan berbahaya ke aplikasi dan memeriksa respons aplikasi tersebut. DAST mensimulasikan serangan dunia nyata dan melihat bagaimana aplikasi bereaksi terhadap input yang tidak valid atau tidak diinginkan.

Dalam konteks secure coding, DAST berperan penting untuk memastikan keamanan aplikasi yang sudah berjalan. Beberapa peran DAST meliputi:

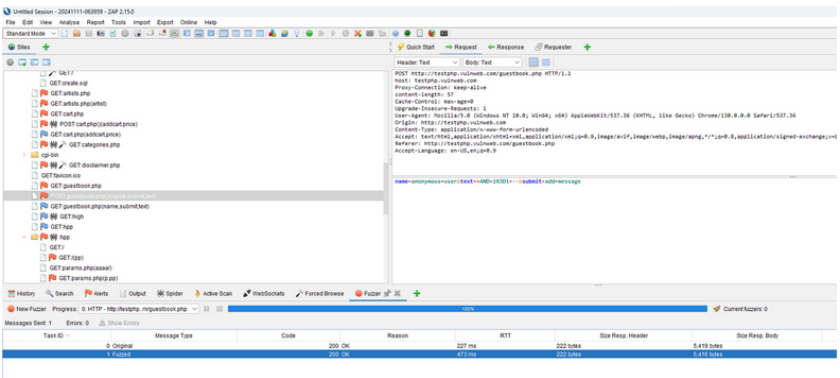
- Mendeteksi Kerentanan: Menemukan masalah keamanan dalam kondisi nyata, seperti serangan injeksi SQL dan XSS, yang tidak terdeteksi dalam pengujian statis.
- Memastikan Validasi Input: Verifikasi mekanisme validasi dan encoding input berfungsi dengan baik untuk mencegah eksploitasi.
- Mengidentifikasi Kebocoran Informasi: Mendeteksi kebocoran informasi sensitif melalui pesan error dan memastikan error handling yang tepat.

- Otomatisasi dalam CI/CD: Mengintegrasikan alat DAST dalam pipeline CI/CD untuk menguji keamanan setiap perubahan aplikasi secara otomatis.

DAST membantu tim pengembangan memperbaiki kerentanan sebelum aplikasi dirilis ke produksi.

Contoh Aplikasi DAST - OWASP ZAP

OWASP ZAP (Zed Attack Proxy) adalah alat keamanan aplikasi open-source yang membantu menemukan kerentanan dalam aplikasi web. Dikembangkan oleh OWASP, ZAP digunakan oleh pengembang dan penguji keamanan untuk mengidentifikasi dan memperbaiki masalah keamanan. Fitur-fitur yang disediakan termasuk pemindaian otomatis, pengujian manual, dan integrasi dengan alat lain. ZAP populer karena kekuatan dan kemudahan penggunaannya, cocok untuk pemula maupun profesional di keamanan aplikasi.



Gambar 5.2 Tampilan Aplikasi OWASP ZAP
(Sumber: www.zaproxy.org)

Untuk mendownload tools OWASP ZAP dapat menggunakan link berikut : https://www.zaproxy.org/

5.1.3. Penetration Testing (Pentest)

Pentesting adalah proses pengujian keamanan aplikasi atau sistem dengan melakukan simulasi serangan oleh penyerang untuk mengidentifikasi dan mengeksploitasi kelemahan keamanan. Dalam secure coding, pentesting bertujuan untuk mengungkapkan celah keamanan dalam kode dan konfigurasi aplikasi sehingga tim pengembang dapat memperbaikinya sebelum terjadi serangan nyata.

Mengapa Pentesting Penting dalam Secure Coding?

- Mengidentifikasi Kerentanan dalam Kode: Pentesting membantu mengidentifikasi kelemahan keamanan, seperti SQL Injection, Cross-Site Scripting (XSS), dan Remote Code Execution (RCE), yang mungkin muncul akibat implementasi kode yang tidak aman.
- Mendeteksi Kerentanan di Lingkungan yang Sebenarnya: Berbeda dengan pengujian statis atau dinamis, pentesting memberikan gambaran bagaimana aplikasi bereaksi terhadap serangan nyata di lingkungan produksi atau hampir produksi.
- Validasi Keamanan Aplikasi: Pentesting memvalidasi keamanan aplikasi dengan menguji bagaimana sistem berperilaku di bawah berbagai jenis serangan, sehingga memberikan gambaran yang jelas mengenai tingkat keamanan aplikasi.

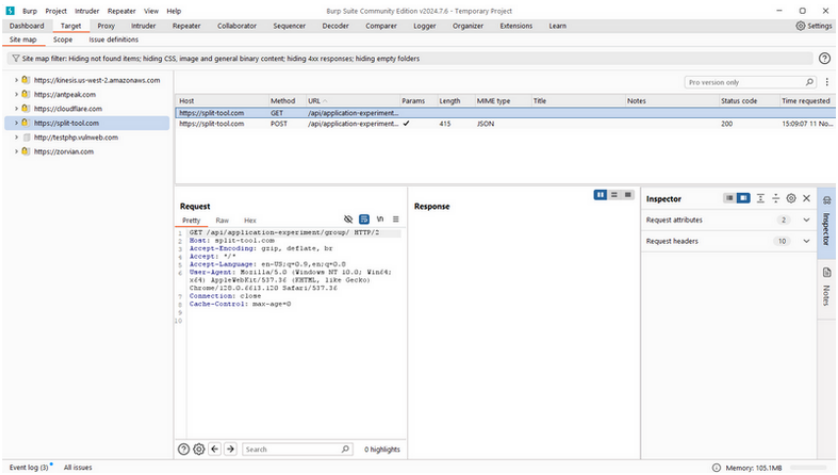
Tools yang Digunakan dalam Pentesting Secure Coding

- Burp Suite: Dapat digunakan untuk memeriksa kerentanan injeksi, XSS, serta untuk memanipulasi permintaan dan respons HTTP.

- OWASP ZAP: Alat open-source yang dapat mengidentifikasi berbagai kerentanan web umum.
- SQLmap: Digunakan khusus untuk mendeteksi dan mengeksploitasi SQL Injection.
- Metasploit: Digunakan untuk eksploitasi dan pengujian berbagai kerentanan aplikasi dan jaringan.
- Nmap: Untuk memeriksa port dan layanan yang terbuka pada server aplikasi, membantu menemukan titik serangan potensial.

Contoh Aplikasi Pentest - Burp Suite

Burp Suite adalah alat yang sangat berguna dalam bidang keamanan siber. Ia digunakan terutamanya untuk menguji keamanan aplikasi web dengan membolehkan pengguna menganalisis lalu lintas HTTP dan HTTPS antara pelayar dan pelayan. Dengan berbagai ciri yang ditawarkan, Burp Suite membolehkan penguji keamanan mengenal pasti kelemahan dalam aplikasi dan membantu dalam memperkuat pertahanan siber. Alat ini amat popular dalam kalangan profesional keamanan kerana kemampuannya yang komprehensif dan fleksibiliti dalam menangani berbagai jenis serangan siber.



Gambar 5.3 Tampilan Aplikasi Burp Suite
(Sumber: https://portswigger.net)

Untuk mendownload tools Burp Suite dapat menggunakan link berikut : https://portswigger.net/burp

5.2. Perbandingan SAST, DAST, dan Pentest

SAST (Static Application Security Testing) adalah metode pengujian keamanan yang menganalisis kode sumber tanpa menjalankannya, membantu menemukan kerentanan awal seperti injeksi SQL atau XSS.

DAST (Dynamic Application Security Testing) dilakukan pada aplikasi yang berjalan, fokus mendeteksi kerentanan yang hanya terlihat saat aplikasi operasional, seperti masalah validasi input dan autentikasi.

Pentesting (Penetration Testing) adalah simulasi serangan untuk mengeksploitasi kerentanan, dilakukan oleh profesional untuk menilai keamanan sistem secara menyeluruh.

Tabel 5.1 Perbandingan SAST, DAST, dan Pentest

Faktor	SAST	DAST	Pen Testing
Waktu Pelaksanaan	Tahap awal (saat coding)	Setelah aplikasi berjalan	Biasanya setelah aplikasi selesai
Deteksi Kerentanan	Kelemahan pada level kode	Masalah saat runtime	Kerentanan dari perspektif penyerang
Jenis Kerentanan	Kesalahan sintaksis dan logika	Input/output handling, XSS	Akses tidak sah, eskalasi hak akses
Alat yang Digunakan	SonarQube, Checkmarx, Veracode	OWASP ZAP, Burp Suite	Metasploit, Wireshark, Kali Linux
Kelebihan	Deteksi awal, hemat biaya	Menyimulasikan lingkungan nyata	Realistis, deteksi mendalam
Kekurangan	Banyak false positives	Tidak memeriksa level kode	Biaya tinggi, waktu eksekusi lama

Ketiga metode ini saling melengkapi dalam strategi keamanan aplikasi yang komprehensif untuk melindungi dari ancaman.

5.3. Metode Fuzzy Testing

Fuzzy testing, atau fuzz testing, adalah metode pengujian perangkat lunak di mana input acak atau tidak terduga (disebut sebagai "fuzz") dimasukkan ke dalam sistem untuk mengidentifikasi kerentanan, kesalahan, atau perilaku yang tidak diinginkan. Fuzz testing sangat efektif dalam mengidentifikasi kelemahan keamanan, seperti buffer overflow, atau kesalahan yang mungkin terjadi ketika aplikasi menerima data yang tidak valid atau tidak diantisipasi.

Fuzzy testing bertujuan untuk:

- Menemukan Kesalahan Input: Memeriksa bagaimana sistem menangani data yang rusak atau tidak terstruktur.
- Mengidentifikasi Kerentanan Keamanan: Menguji kerentanan yang bisa dieksploitasi oleh penyerang, seperti SQL Injection atau serangan Cross-Site Scripting (XSS).
- Memastikan Stabilitas: Memastikan aplikasi tetap stabil dan tidak crash saat menerima input yang tidak sesuai.

Pengujian ini dilakukan dengan memberi input acak pada aplikasi, dan jika aplikasi menampilkan perilaku yang tidak diharapkan (misalnya, crash atau pengecualian tidak tertangani), maka ada potensi kelemahan yang perlu diperbaiki.

Cara Menjalankan Fuzz Testing

- Perhatikan Output: Perhatikan apakah aplikasi menampilkan hasil yang tidak wajar atau error.
- Tinjau Error Handling: Pastikan setiap error yang muncul ditangani dengan benar atau diantisipasi oleh fungsi yang diuji.

- Periksa Kelemahan: Jika ditemukan error, tinjau kembali fungsi untuk melihat apakah ada bagian kode yang harus diperbaiki, misalnya penanganan input yang lebih baik atau validasi tambahan.

Untuk memahami lebih dalam tentang fuzzy testing, berikut ini contoh kasus untuk fungsi login. Misalkan kita memiliki fungsi login yang memeriksa apakah username dan password valid. Berikut adalah implementasi dasar tanpa fuzz testing:

```
def login(username, password):
    if username == "admin" and password == "securepassword":
        return "Login successful"
    else:
        return "Login failed"
```

Sekarang kita akan membuat fungsi fuzz testing yang menguji input acak pada fungsi login.

```
import random
import string

def fuzz_login_test():
    for _ in range(100):  # Lakukan 100 kali pengujian dengan input acak
        # Buat username dan password acak
        fuzz_username = ''.join(random.choices(string.ascii_letters + string.digits, k=8))
        fuzz_password = ''.join(random.choices(string.ascii_letters + string.digits, k=8))

        # Cetak input yang sedang diuji
        print(f"Testing with username: {fuzz_username} and password: {fuzz_password}")

        try:
            result = login(fuzz_username, fuzz_password)
            print(f"Result: {result}")
        except Exception as e:
            print(f"Error encountered with input ({fuzz_username}, {fuzz_password}): {e}")

# Jalankan fuzz testing
fuzz_login_test()
```

5.4. Metode Input Validation Testing

Input validation testing adalah proses pengujian yang bertujuan untuk memastikan bahwa aplikasi hanya menerima data yang benar dan sesuai. Pengujian ini bertujuan untuk mencegah terjadinya input berbahaya yang bisa mengganggu atau mengeksploitasi sistem. Input validation sering kali diuji untuk mencegah berbagai serangan, seperti SQL Injection, Cross-Site Scripting (XSS), dan buffer overflow. Teknik ini berfungsi untuk memastikan bahwa aplikasi aman dari data yang tidak valid atau mencurigakan yang dapat menyebabkan kerentanan atau bahkan akses tidak sah.

Proses input validasi testing umumnya melibatkan beberapa langkah penting. Pertama, identifikasi jenis data yang akan diuji, seperti teks, angka, atau tanggal. Selanjutnya, tentukan aturan validasi, seperti format yang diharapkan, batasan nilai minimum atau maksimum, serta pengecualian atau kondisi khusus lainnya. Setelah aturan ditetapkan, buat skenario pengujian yang mencakup berbagai kemungkinan input, baik yang valid maupun yang tidak valid.

Kemudian, lakukan pengujian dengan memasukkan data ke dalam sistem dan amati bagaimana sistem merespons setiap jenis input. Pastikan bahwa sistem menolak input yang tidak valid dengan memberikan pesan kesalahan yang jelas dan membantu. Setelah itu, evaluasi hasil pengujian untuk memastikan bahwa semua aturan validasi telah diterapkan dengan benar. Jika ditemukan kesalahan atau kekurangan, lakukan perbaikan dan ulangi pengujian hingga sistem berfungsi sesuai harapan.

Terakhir, dokumentasikan seluruh proses dan hasil pengujian untuk referensi di masa depan, serta untuk memastikan bahwa

semua anggota tim memahami bagaimana validasi dilakukan dan mengapa langkah-langkah tersebut penting bagi kualitas sistem.

Berikut ini contoh kasus kode tanpa menggunakan validasi:

```
<?php
// file: user_info.php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "test_db";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// Mengambil parameter `id` dari URL
$id = $_GET['id'];
// Query tanpa validasi input (vulnerable)
$sql = "SELECT * FROM users WHERE id = $id";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
while ($row = $result->fetch_assoc()) {
echo "ID: " . $row["id"] . " - Name: " . $row["name"] . "<br>";
}
} else {
echo "0 results";
}
$conn->close();
?>
```

Save file diatas dengan nama user_info.php. Kemudian kita buat database dengan kueri sql sebagai berikut :

```
CREATE DATABASE test_db;
USE test_db;
CREATE TABLE users (
id INT PRIMARY KEY,
name VARCHAR(50)
pass VARCHAR(30)
);
INSERT INTO users (id, name, pass) VALUES (1, 'Alice', ‘123’),
(2, 'Bob', ‘admin123’), (3, 'Fulan', ‘zonk);
```

Setelah itu, kita akan buatkan kode versi amannya sebagai berikut:

```
<?php
// file: user_info_secure.php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "test_db";
$conn = new mysqli($servername, $username, $password,
$dbname);
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// Mengambil parameter `id` dari URL
$id = $_GET['id'];
// Validasi input dan prepared statement
if (!is_numeric($id)) {
die("Invalid ID");
}
$stmt = $conn->prepare("SELECT * FROM users WHERE id =
?");
$stmt->bind_param("i", $id);
$stmt->execute();
```

```
$result = $stmt->get_result();
if ($result->num_rows > 0) {
while ($row = $result->fetch_assoc()) {
echo "ID: " . $row["id"] . " - Name: " . $row["name"] . "<br>";
}
} else {
echo "0 results";
}
$stmt->close();
$conn->close();
?>
```

Save file diatas dengan nama user_info_secure.php. Kemudian silahkan pembaca bandingkan hasilnya dari 2 kode php tersebut.

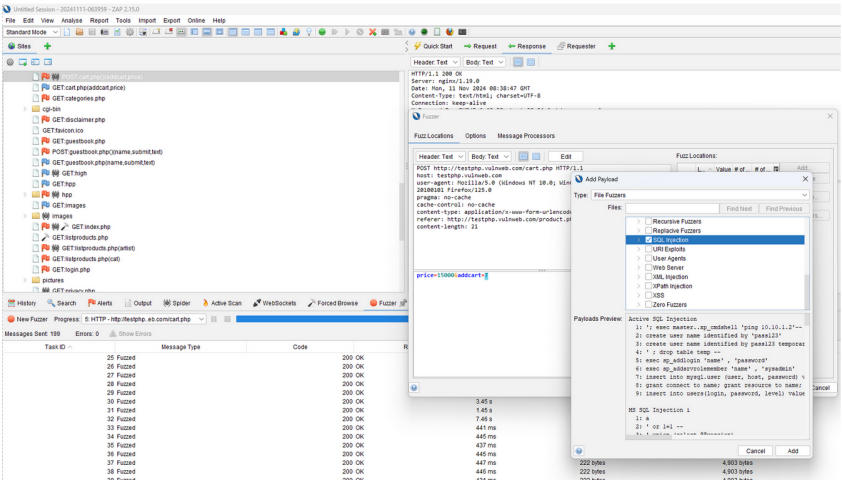
5.5. Studi Kasus

Jika sudah paham konsep dari Fuzzy Testing dan Input Validation Testing, sekarang kita akan mencoba untuk melakukan pengujian SQL Injection dengan menggunakan metode Fuzzy Testing.

Jika fungsi login melibatkan database, maka fuzz testing dapat mendeteksi potensi kerentanan SQL Injection. Contohnya, jika input tidak divalidasi dengan baik, penyerang bisa saja menyisipkan karakter khusus dalam username atau password untuk mencoba mengeksploitasi database.

Dengan fuzz testing, input seperti username = "admin' OR '1'='1" akan mengungkapkan apakah aplikasi rentan terhadap serangan ini, dan membantu pengembang membuat validasi yang lebih kuat.

Gunakan Tools OWASP ZAP untuk melakukan pengujian fuzz terhadap website http://testphp.vulnweb.com dan https://testportal.helium.sh/home.php



Gambar 5.4 Penggunaan Fuzzy Testing pada OWASP ZAP

CHAPTER 6
SECURE SOFTWARE DEPLOYMENT

Materi:

- Konfigurasi server dan lingkungan yang aman.
- Proses hardening aplikasi dan server.
- Menggunakan pipeline CI/CD yang aman.

Durasi:

- 1 Jam

Tujuan:

- Mengamankan aplikasi saat proses deployment dan memastikan lingkungan yang aman.

Target Pembaca:

- Karyawan IT (SysAdmin, Network Engineer, DevOps)

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

6.1. Konfigurasi Server dan Lingkungan Yang Aman

Konfigurasi ini bertujuan melindungi perangkat lunak dan server dari ancaman setelah peluncuran aplikasi. Langkah-langkah keamanan seperti pembaruan rutin, pemantauan aktivitas mencurigakan, dan penggunaan firewall yang kuat dapat mengurangi risiko serangan siber. Melibatkan tim keamanan dalam pengembangan dan peluncuran aplikasi serta melatih karyawan tentang praktik keamanan terbaik juga penting. Dengan cara ini, perusahaan dapat memberikan layanan yang lebih aman dan melindungi aset digital dari eksploitasi.

6.1.1. Pembaruan Sistem dan Patch Secara Berkala

- Server harus selalu menggunakan versi terbaru dari sistem operasi dan perangkat lunak lain yang dijalankan, termasuk database, framework, dan server aplikasi.
- Patch keamanan harus diterapkan sesegera mungkin untuk menutup celah yang dapat dieksploitasi oleh penyerang.
- Contoh: Menjalankan sudo apt update && sudo apt upgrade secara berkala pada server berbasis Linux atau menggunakan solusi otomatis seperti Ansible atau Puppet untuk mengelola patch di banyak server sekaligus.

6.1.2. Minimalkan Permukaan Serangan (Attack Surface)

- Hanya aplikasi dan layanan yang diperlukan untuk menjalankan aplikasi yang harus diaktifkan. Menonaktifkan layanan atau port yang tidak diperlukan mengurangi peluang bagi penyerang untuk mengeksploitasi sistem.
- Contoh: Jika aplikasi Anda tidak menggunakan FTP atau SMTP, layanan ini dapat dimatikan agar tidak terbuka pada port jaringan.

6.1.3. Gunakan Firewall dan Pembatasan Akses Jaringan

- Gunakan firewall untuk mengontrol lalu lintas masuk dan keluar dari server. Hanya izinkan akses ke port dan layanan yang benar-benar diperlukan, seperti membuka port 80 dan 443 untuk HTTP/HTTPS pada server web.
- Pertimbangkan juga untuk mengonfigurasi IP-based Access Control Lists (ACLs) untuk membatasi akses berdasarkan alamat IP.
- Contoh: Menggunakan iptables atau UFW di Linux untuk mengonfigurasi firewall, atau firewall aplikasi cloud yang disediakan oleh penyedia seperti AWS Security Groups atau Azure Network Security Groups.

6.1.4. Konfigurasi Hak Akses dan Privilege Management

- Terapkan prinsip least privilege dalam konfigurasi hak akses pengguna dan peran. Hanya pengguna yang berwenang dan membutuhkan akses yang boleh mengakses server atau layanan tertentu.
- Gunakan akses berbasis peran (Role-Based Access Control, RBAC) untuk mengontrol siapa yang dapat mengakses atau mengelola aplikasi pada server.
- Contoh: Menggunakan sudo di Linux untuk memberikan akses terbatas atau menggunakan peran IAM (Identity and Access Management) di cloud untuk membatasi hak akses pengguna.

6.1.5. Amankan Koneksi Jaringan

- Gunakan protokol yang aman seperti HTTPS, SSH, dan SFTP. Hindari penggunaan protokol yang tidak terenkripsi seperti HTTP atau FTP untuk menghindari pencurian data.

- Terapkan enkripsi pada data yang dikirim melalui jaringan, terutama data sensitif, untuk melindungi dari serangan man-in-the-middle.
- Contoh: Menggunakan sertifikat SSL/TLS untuk semua koneksi ke server aplikasi dan database, serta mengonfigurasi SSH dengan kunci autentikasi daripada password untuk akses server.

6.1.6. Enkripsi Data Sensitif

- Data sensitif harus dienkripsi baik saat dalam perjalanan (in transit) maupun saat diam (at rest).
- Pertimbangkan untuk mengenkripsi disk atau partisi tertentu, terutama yang menyimpan data sensitif atau kredensial.
- Contoh: Menggunakan enkripsi disk seperti dm-crypt atau LUKS pada Linux, dan menerapkan HTTPS untuk melindungi data saat ditransmisikan melalui jaringan.

6.1.7. Logging dan Monitoring

- Aktifkan logging pada server untuk mencatat aktivitas yang mencurigakan atau penting. Hal ini mencakup pencatatan akses, modifikasi, atau aktivitas yang gagal pada server.
- Monitoring server dapat membantu dalam mendeteksi anomali atau serangan dengan cepat, memungkinkan tim respons untuk bertindak segera.
- Contoh: Gunakan syslog atau layanan log berbasis cloud, serta alat monitoring seperti Nagios, Zabbix, atau solusi cloud untuk memantau performa dan keamanan.

6.1.8. Konfigurasi Otentikasi Multi-Faktor (MFA)

- Jika memungkinkan, gunakan otentikasi multi-faktor untuk menambahkan lapisan keamanan tambahan pada akses administrasi ke server.
- MFA dapat mencegah akses tidak sah, bahkan jika kata sandi pengguna telah dicuri.
- Contoh: Menyediakan MFA untuk login administrator melalui layanan seperti Google Authenticator atau solusi MFA dari penyedia cloud seperti AWS atau Azure.

6.1.9. Backup dan Disaster Recovery

- Pastikan Anda memiliki proses backup data secara berkala, serta rencana pemulihan jika terjadi insiden seperti serangan ransomware atau bencana alam.
- Contoh: Membuat backup harian atau mingguan yang disimpan di lokasi yang berbeda, seperti di cloud atau perangkat penyimpanan offline.

6.1.10. Validasi dan Konfigurasi File Permissions

- Konfigurasi hak akses file yang ketat pada server untuk membatasi pengguna atau proses yang tidak diinginkan dalam mengakses atau memodifikasi file tertentu.
- Pastikan bahwa hanya pengguna atau proses yang sah yang memiliki akses ke file konfigurasi sensitif, seperti file environment (.env) yang berisi kredensial aplikasi.
- Contoh: Menetapkan izin file chmod 600 pada file konfigurasi sensitif dan menjalankan aplikasi di bawah pengguna khusus (misalnya, pengguna www-data di Nginx atau Apache).

Dengan melakukan konfigurasi-konfigurasi di atas, server akan lebih terlindungi dari ancaman dan serangan yang mungkin

terjadi. Hal ini sangat penting dalam tahap deployment untuk memastikan lingkungan yang aman bagi aplikasi dan data pengguna.

6.2. Hardening Aplikasi dan Server

Hardening adalah upaya memperkuat aplikasi dan server untuk mengurangi kemungkinan risiko atau kerentanan keamanan, dengan cara menonaktifkan fitur yang tidak diperlukan, mengamankan konfigurasi, serta menerapkan langkah-langkah keamanan tambahan. Berikut adalah tabel yang membandingkan tindakan hardening untuk aplikasi dan server. Tabel 6.1 ini mencakup langkah-langkah penting yang bisa dilakukan untuk mengamankan aplikasi dan server secara lebih mendalam.

Tabel 6.1 Langkah Penting Mengamankan Aplikasi & Server

Kategori	Hardening Aplikasi	Hardening Server
Menghapus Komponen Tidak Diperlukan	Hapus modul atau plugin yang tidak digunakan untuk mencegah celah keamanan.	Nonaktifkan layanan atau protokol yang tidak diperlukan seperti FTP, Telnet, atau SMB.
Pembaruan dan Patch	Pastikan aplikasi selalu diperbarui dan menggunakan versi library terbaru untuk menutup kerentanan.	Terapkan pembaruan OS dan perangkat lunak secara berkala, otomatisasi patch jika memungkinkan.
Keamanan Jaringan	Terapkan firewall aplikasi (WAF) untuk memfilter dan	Gunakan firewall server (misalnya iptables, UFW) untuk

	memonitor permintaan yang mencurigakan.	membatasi akses pada port tertentu.
Izin dan Akses Kontrol	Terapkan prinsip least privilege, atur izin pengguna sesuai kebutuhan fungsional.	Konfigurasi izin akses file yang ketat di sistem (misalnya chmod, chown) dan gunakan ACL.
Proteksi Data Sensitif	Enkripsi data sensitif (misalnya, kata sandi, informasi pribadi) dengan algoritma yang kuat.	Aktifkan enkripsi pada disk atau partisi untuk melindungi data jika server hilang atau dicuri.
Konfigurasi Logging	Gunakan logging aplikasi (misalnya, Log4j) untuk melacak kejadian yang signifikan.	Aktifkan logging sistem (syslog) dan pantau aktivitas untuk mendeteksi anomali.
Validasi Input dan Output	Terapkan validasi input yang ketat dan gunakan encoding untuk output yang aman dari XSS.	Tidak berlaku langsung; namun, monitoring data yang masuk dari jaringan eksternal sangat penting.
Enkripsi Data di Transit	Gunakan HTTPS/TLS untuk mengamankan transmisi data antara klien dan server.	Terapkan sertifikat SSL pada layanan server, dan gunakan VPN untuk akses internal yang aman.

Proteksi Serangan Aplikasi	Gunakan mekanisme pencegahan injeksi, seperti query parameterized SQL untuk menghindari SQL Injection.	Gunakan IDS/IPS untuk memantau lalu lintas dan mendeteksi aktivitas mencurigakan.
Autentikasi dan MFA	Implementasikan autentikasi dua faktor (2FA) untuk akses admin aplikasi.	Terapkan MFA untuk SSH atau akses administratif ke server.
Backup dan Recovery	Backup rutin data aplikasi dan simpan secara aman untuk pemulihan saat terjadi kerusakan.	Backup sistem secara berkala, serta enkripsi backup yang disimpan di lokasi terpisah.
Pengamanan Error Handling	Sembunyikan informasi sensitif dalam pesan error yang dapat dimanfaatkan oleh penyerang.	Nonaktifkan tampilan error server yang detail dan batasi informasi pada log error yang sensitif.

Dengan mengikuti panduan hardening ini, sistem akan menjadi lebih aman dan tahan terhadap berbagai ancaman siber. Langkah-langkah yang dijelaskan mencakup penutupan port yang tidak perlu, pembaruan rutin perangkat lunak, dan penerapan kebijakan kata sandi yang kuat. Selain itu, penting untuk selalu memonitor aktivitas jaringan dan menerapkan enkripsi pada data sensitif. Dengan demikian, tidak hanya menjaga integritas sistem, tetapi juga melindungi informasi berharga dari akses yang tidak sah.

CHAPTER 7
STRATEGI PENERAPAN SECURE SDLC

Materi:

- Pentingnya SSDL dalam Software Startup
- Model Strategi Penerapan SSDL di Software Startup
- Kesimpulan.

Durasi:

- 1 Jam

Tujuan:

- Memahami tantangan dan keunikan penerapan Secure SDLC dan Mengetahui langkah-langkah dan prioritas penerapan keamanan yang efektif pada software Startup.

Target Pembaca:

- Top Level Management (C-Level, CTO, CISO)

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

7.1. Pentingnya SSDL Dalam Startup

Secure Software Development Lifecycle (SSDL) adalah pendekatan dalam pengembangan perangkat lunak yang memasukkan praktik keamanan di setiap tahap pengembangan, mulai dari perencanaan hingga pemeliharaan. Tujuan SSDL adalah untuk mengidentifikasi dan mengatasi potensi kerentanan keamanan sebelum perangkat lunak dirilis, sehingga lebih sulit bagi penyerang untuk mengeksploitasi sistem.

SSDL bukan hanya tentang menambahkan keamanan di akhir pengembangan, tetapi merupakan pendekatan berlapis yang memastikan setiap aspek dari kode, desain, hingga penerapan sudah dipertimbangkan dari sisi keamanan. Hal ini melibatkan langkah-langkah yang memastikan perangkat lunak terlindungi dari ancaman internal dan eksternal.

Startup seringkali memiliki fokus utama pada kecepatan pengembangan untuk bisa cepat masuk ke pasar. Namun, jika keamanan diabaikan, risiko yang dihadapi startup dapat berdampak serius, baik secara finansial maupun reputasi.

Beberapa alasan mengapa SSDL sangat penting bagi startup:

- Melindungi Data Pengguna: Startup sering mengelola data sensitif pengguna. Tanpa keamanan yang memadai, data ini rentan terhadap kebocoran atau penyalahgunaan.
- Menghemat Biaya Jangka Panjang: Dengan mengidentifikasi kerentanan sejak awal, startup dapat mengurangi biaya untuk memperbaiki kesalahan di tahap akhir atau setelah perangkat lunak dirilis.
- Membangun Kepercayaan Pengguna: Keamanan yang kuat meningkatkan kepercayaan pengguna, yang sangat

penting untuk startup dalam menarik dan mempertahankan pelanggan.

- Mematuhi Regulasi dan Standar Industri: Beberapa industri mengharuskan standar keamanan tertentu dalam perangkat lunak. Penerapan SSDL membantu startup mematuhi persyaratan hukum dan regulasi yang relevan.

7.1.1. Risiko Keamanan di Lingkungan Startup

Lingkungan startup biasanya memiliki karakteristik unik yang membuat keamanan menjadi tantangan. Beberapa risiko yang umum dihadapi oleh startup yang belum menerapkan SSDL adalah:

- Keterbatasan Sumber Daya:
    - Banyak startup memiliki keterbatasan anggaran dan tenaga ahli, sehingga alokasi untuk keamanan sering kali diabaikan.
    - Solusi: Menggunakan alat open-source dan menerapkan praktik secure coding yang tidak membutuhkan biaya besar.
- Fokus pada Kecepatan Pengembangan:
    - Startup berfokus pada inovasi dan mempercepat rilis produk (Minimum Viable Product, atau MVP). Ini kadang membuat keamanan menjadi prioritas kedua.
    - Solusi: Mengintegrasikan praktik SSDL dalam alur kerja pengembangan yang ada, misalnya dengan melakukan code review dan penetration testing sederhana.
- Kurangnya Kesadaran Keamanan:

    - Karena fokus pada pengembangan dan inovasi, banyak tim di startup belum memiliki pemahaman mendalam tentang keamanan perangkat lunak.
    - Solusi: Melakukan pelatihan atau program security awareness sederhana bagi tim, terutama developer dan product manager.
- Risiko Serangan Eksternal:
    - Startup, meskipun kecil, tetap menjadi target serangan siber karena banyak yang memiliki data pengguna yang berharga.
    - Solusi: Menggunakan firewall dan Intrusion Detection System (IDS) dasar yang mampu mendeteksi aktivitas yang mencurigakan.

7.1.2. Komponen Penting dalam SSDL

SSDL memiliki beberapa komponen inti yang dapat diadaptasi dalam lingkungan startup:

- Awareness dan Training Keamanan
    - Semua tim harus diberikan pemahaman dasar tentang keamanan perangkat lunak, seperti pentingnya enkripsi, otentikasi, dan validasi input.
    - Training yang berfokus pada secure coding practices sangat membantu dalam mengurangi kesalahan keamanan dalam pengembangan.
- Identifikasi dan Pengelolaan Risiko
    - Identifikasi risiko keamanan pada awal proyek dan lakukan analisis risiko sederhana untuk menentukan prioritas keamanan.

    - Proses ini mencakup pemetaan kerentanan potensial dan strategi mitigasinya sesuai dengan sumber daya yang tersedia.

- Automasi Keamanan dalam CI/CD Pipeline
    - Menggunakan alat otomatis dalam pipeline CI/CD untuk memeriksa kerentanan pada kode sebelum dideploy ke produksi.
    - Alat-alat seperti OWASP ZAP untuk Dynamic Application Security Testing (DAST) atau SonarQube untuk Static Application Security Testing (SAST) dapat diintegrasikan dengan pipeline yang ada.
- Pengujian Keamanan Secara Berkala
    - Menjalankan pengujian keamanan secara berkala pada setiap rilis perangkat lunak atau ketika ada perubahan besar pada kode.
    - Startup dapat melakukan pengujian sederhana seperti fuzz testing untuk mendeteksi respons aplikasi terhadap input yang tidak terduga, atau penetration testing yang lebih mendalam.

7.1.3. Karakteristik Startup

Startup memiliki karakteristik unik yang memengaruhi bagaimana SSDL diterapkan. Beberapa karakteristik tersebut adalah:

- Fokus pada Kecepatan dan Inovasi: Startup biasanya bekerja dalam lingkungan yang dinamis dan berusaha untuk cepat meluncurkan produk baru atau Minimum Viable Product (MVP) untuk menarik perhatian pengguna dan investor. Fokus ini kadang membuat keamanan menjadi prioritas kedua atau kurang diperhatikan.

- Keterbatasan Anggaran dan Sumber Daya: Startup sering kali memiliki anggaran terbatas untuk keamanan dan jumlah tim yang kecil, sehingga sulit untuk mengalokasikan tenaga kerja atau investasi yang besar dalam keamanan perangkat lunak.
- Skalabilitas dan Perubahan Cepat: Banyak startup berkembang pesat dan membutuhkan solusi keamanan yang dapat dengan cepat beradaptasi dengan perubahan dalam tim, infrastruktur, atau teknologi.

Dengan adanya karakteristik ini, startup membutuhkan pendekatan SSDL yang fleksibel, efektif, dan efisien dalam hal biaya.

7.2. Model Strategi Penerapan SSDL di Startup

Kerangka konseptual ini menyatakan bahwa adopsi Secure Software Development Lifecycle (SSDL) di startup perangkat lunak tidak hanya ditentukan oleh aspek teknis dari praktik pengembangan perangkat lunak yang aman itu sendiri. Sebaliknya, penerapan SSDL dipengaruhi oleh interaksi kompleks dari berbagai faktor, yang meliputi karakteristik khas startup dan ekosistem yang lebih luas di mana mereka beroperasi.

Secara spesifik, kerangka ini menyoroti bagaimana keterbatasan sumber daya, kecepatan pengembangan, toleransi terhadap risiko, dan tingkat kesadaran keamanan—semua karakteristik yang umumnya dimiliki oleh startup—dapat menjadi faktor yang menghambat atau justru memfasilitasi penerapan praktik SSDL. Selain itu, faktor eksternal seperti kebijakan pemerintah yang mendukung keamanan siber, ketersediaan pendanaan dan program pendampingan, serta keberadaan keahlian keamanan siber dalam ekosistem juga memainkan peran penting dalam membentuk persepsi kebutuhan serta kelayakan implementasi SSDL.

Penelitian ini bertujuan untuk mengkaji secara empiris hubungan-hubungan tersebut dalam konteks startup perangkat lunak di Jawa Barat, Indonesia. Melalui studi ini, diharapkan akan diperoleh pemahaman yang lebih mendalam tentang tantangan spesifik serta faktor-faktor pendukung yang berperan dalam proses adopsi SSDL di dalam ekosistem startup yang unik ini.

7.2.1. Tantangan Utama Terhadap Startup

Tantangan paling umum yang diidentifikasi oleh responden adalah kurangnya kesadaran mengenai pentingnya SSDL. Hal ini menunjukkan adanya kebutuhan mendesak untuk inisiatif edukasi dan kampanye kesadaran guna menekankan pentingnya praktik secure coding dan manfaat dari penerapan siklus pengembangan yang berfokus pada keamanan. Tantangan utama kedua yang banyak disebutkan adalah keterbatasan anggaran. Hal ini menunjukkan bahwa organisasi mungkin mengalami kesulitan dalam mengalokasikan sumber daya yang memadai untuk mendukung penerapan SSDL, yang dapat berdampak pada kemampuan mereka untuk berinvestasi dalam alat, pelatihan, dan tenaga kerja yang diperlukan. Hambatan teknis juga menjadi kendala signifikan. Hambatan teknis ini mencakup berbagai masalah, seperti kurangnya keahlian dalam praktik secure coding, kesulitan dalam mengintegrasikan alat keamanan ke dalam alur kerja yang sudah ada, atau tantangan dalam beradaptasi dengan ancaman keamanan dan kerentanan yang terus berkembang. Keterbatasan waktu juga sering disebutkan. Hal ini menunjukkan bahwa organisasi mungkin kesulitan menyeimbangkan kebutuhan untuk menerapkan praktik pengembangan yang aman dengan tekanan untuk menghadirkan produk dan pembaruan perangkat lunak dalam tenggat waktu yang ketat. Terakhir, kekurangan sumber daya manusia yang terlatih. Hal ini menegaskan pentingnya investasi dalam program pelatihan dan pengembangan untuk membekali tim pengembangan perangkat lunak dengan

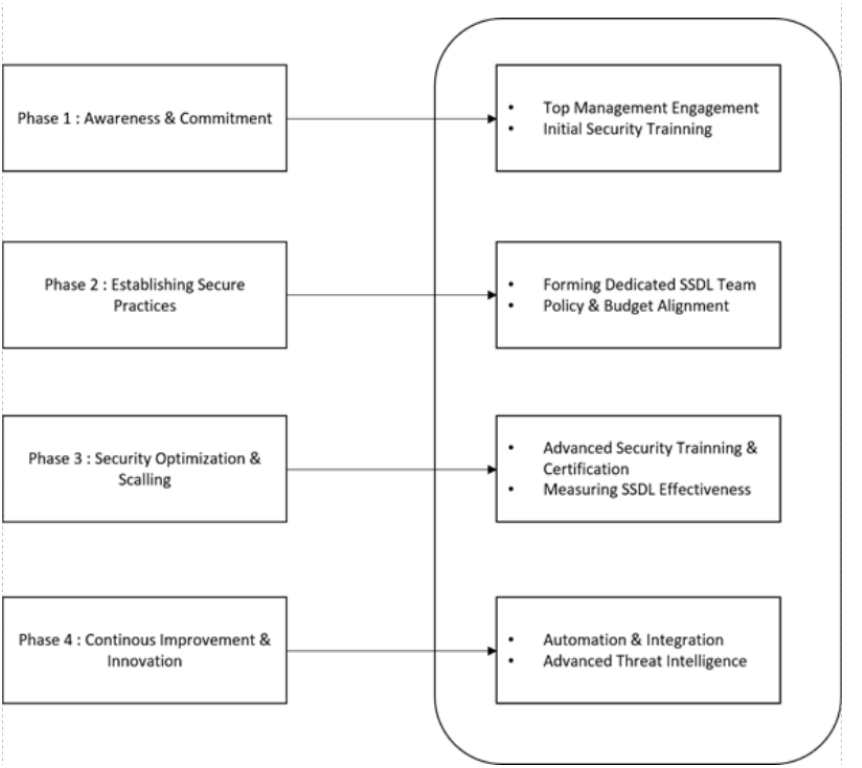
keterampilan dan pengetahuan yang diperlukan guna menerapkan praktik SSDL secara efektif.

7.2.2. Model Strategis Penerapan SSDL

Model strategis yang diusulkan untuk penerapan secure software development lifecycle (SSDL) pada startup perangkat lunak menggunakan kerangka tata kelola keamanan informasi untuk mengatasi tantangan khusus yang dihadapi oleh organisasi-organisasi ini, seperti yang ditampilkan pada Gambar 7.1.

Komponen pertama dari model ini adalah pembentukan kerangka tata kelola keamanan informasi yang kuat, yang selaras dengan tujuan dan prioritas bisnis startup secara keseluruhan. Kerangka ini dirancang untuk menentukan peran, tanggung jawab, dan proses pengambilan keputusan yang terkait dengan keamanan informasi di dalam organisasi. Dengan demikian, setiap pertimbangan keamanan akan tertanam dalam operasional dan praktik pengembangan di startup, memastikan keamanan menjadi bagian integral dari setiap proses yang ada.

Komponen kedua dari model ini adalah integrasi langkah-langkah keamanan yang komprehensif ke dalam setiap fase siklus hidup pengembangan perangkat lunak, mulai dari pengumpulan kebutuhan hingga deployment dan pemeliharaan. Langkah-langkah keamanan ini dapat mencakup implementasi praktik secure coding, pengujian keamanan, dan manajemen kerentanan, bersama dengan kegiatan-kegiatan lain yang berkaitan dengan keamanan. Dengan adanya langkah-langkah ini, setiap fase dalam siklus pengembangan diharapkan sudah mencakup kontrol dan langkah antisipasi yang memadai untuk melindungi perangkat lunak dari ancaman.



Gambar 7.1 Model Strategis Penerapan SSDL

Komponen ketiga adalah penerapan proses pemantauan dan peningkatan berkelanjutan untuk SSDL. Proses ini melibatkan peninjauan secara berkala terhadap efektivitas langkah-langkah keamanan yang telah diterapkan, termasuk memasukkan umpan balik dari audit keamanan dan penetration testing. Selain itu, proses ini juga mencakup penyesuaian siklus pengembangan perangkat lunak untuk menghadapi ancaman baru dan memenuhi persyaratan bisnis yang terus berkembang.

7.3. Kesimpulan

Penelitian ini menyajikan model strategis untuk menerapkan siklus hidup pengembangan perangkat lunak yang aman (SSDL)

dalam konteks startup perangkat lunak, dengan memanfaatkan kerangka tata kelola keamanan informasi. Komponen kunci dari model yang diusulkan meliputi pembentukan kerangka tata kelola keamanan informasi, integrasi langkah-langkah keamanan yang komprehensif ke dalam siklus hidup pengembangan perangkat lunak, serta penerapan proses pemantauan dan perbaikan yang berkelanjutan.

Namun, penelitian ini tidak hanya berhenti pada identifikasi hambatan-hambatan tersebut. Penulis menyajikan serangkaian rekomendasi praktis dan dapat dilaksanakan yang dirancang khusus untuk lingkungan startup. Dengan mengadopsi kombinasi inisiatif pendidikan, alokasi sumber daya kreatif, strategi implementasi bertahap, dan budaya pembelajaran berkelanjutan, startup dapat menghadapi tantangan ini dan membangun fondasi keamanan yang kokoh. Implementasi SSDL yang sukses bukan hanya merupakan upaya teknis; ini memerlukan perubahan mendasar dalam pola pikir. Startup yang memprioritaskan keamanan sejak awal, mengintegrasikannya secara mulus ke dalam proses pengembangan, dan mendorong budaya tanggung jawab bersama, akan meraih manfaat jangka panjang yang signifikan. Manfaat ini meliputi peningkatan kepercayaan pelanggan, pengurangan risiko pelanggaran data yang mahal, serta pada akhirnya, bisnis yang lebih kuat dan berkelanjutan.

Para pembuat kebijakan memainkan peran penting dalam mempromosikan adopsi SSDL secara luas di kalangan startup perangkat lunak. Hal ini dapat dicapai melalui peluncuran program dan kampanye yang meningkatkan kesadaran tentang pentingnya SSDL dan manfaatnya bagi keamanan produk serta reputasi perusahaan. Selain itu, memfasilitasi akses mudah ke sumber daya SSDL, seperti panduan praktis, kerangka kerja terstruktur, dan alat yang relevan, akan sangat bermanfaat bagi startup, terutama pada tahap awal pengembangan. Menawarkan

insentif, seperti pemotongan pajak atau hibah, kepada startup yang berkomitmen untuk menerapkan praktik SSDL yang kuat dapat berfungsi sebagai stimulus tambahan. Dari sudut pandang regulasi, menetapkan standar dan peraturan SSDL yang jelas dan mudah diterapkan, yang relevan dengan konteks industri perangkat lunak, akan menciptakan fondasi yang solid. Terakhir, mendorong kolaborasi dan berbagi praktik terbaik SSDL antara startup, akademisi, dan lembaga pemerintah akan mengembangkan ekosistem saling mendukung yang mempercepat pembelajaran kolektif.

Dengan mengadopsi model ini, startup perangkat lunak dapat mengembangkan dan mempertahankan aplikasi perangkat lunak yang aman, yang melindungi kerahasiaan, integritas, dan ketersediaan sistem serta data mereka, sekaligus mengakomodasi kendala operasional dan prioritas bisnis yang unik. Penelitian ini berfungsi sebagai panduan praktis dan seruan untuk bertindak bagi startup perangkat lunak agar memprioritaskan keamanan sebagai nilai inti, memberdayakan mereka untuk membangun perangkat lunak inovatif dan aman untuk masa depan digital yang lebih aman.

CHAPTER 8
DASAR CYBER SECURITY

Materi:

- Apa itu cyber security?
- Jenis-jenis ancaman siber (malware, phishing, ransomware)
- Dampak pelanggaran keamanan data bagi perusahaan.

Durasi:

- 1 Jam

Tujuan:

- Memahami konsep dasar keamanan informasi dan ancaman dalam dunia siber.

Target Pembaca:

- Mahasiswa & Karyawan Non-IT

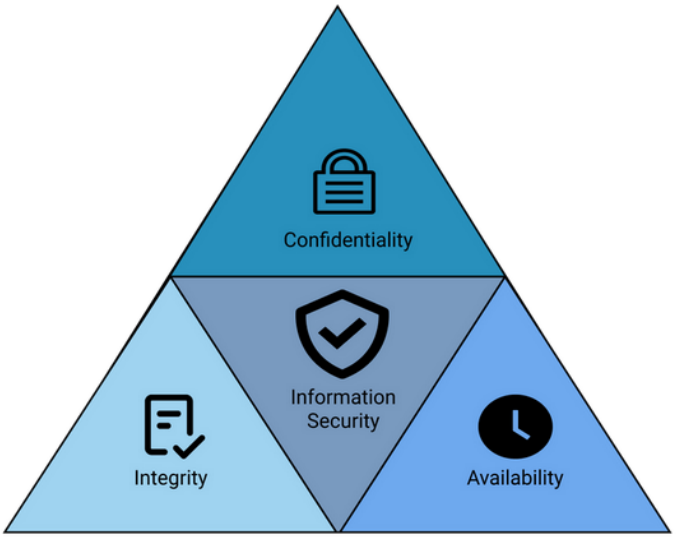
Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

8.1. Pendahuluan Cyber Security

Cybersecurity adalah praktik melindungi sistem, jaringan, dan program dari serangan digital. Serangan siber ini biasanya bertujuan untuk mengakses, mengubah, atau merusak informasi sensitif, mengganggu proses bisnis, atau meminta uang tebusan. Dengan meningkatnya digitalisasi, ancaman siber pun semakin berkembang dan menjadi perhatian utama dalam operasional IT. Oleh karena itu, penting bagi organisasi dan individu untuk mengadopsi langkah-langkah keamanan yang proaktif. Beberapa langkah yang dapat diambil termasuk pembaruan perangkat lunak secara teratur, penggunaan kata sandi yang kuat dan unik, serta implementasi solusi keamanan seperti firewall dan antivirus. Selain itu, pelatihan kesadaran keamanan siber bagi karyawan juga sangat penting untuk mengurangi risiko kesalahan manusia yang dapat dimanfaatkan oleh penyerang. Dengan pendekatan yang komprehensif dan terus berkembang, kita dapat lebih siap menghadapi ancaman siber dan melindungi aset digital kita.

Untuk menjaga keamanan informasi dan sistem yang ada di perusahaan, maka ada acuan sebuah model yang dapat kita gunakan yaitu CIA Triad. CIA Triad adalah model dasar dalam keamanan informasi yang berfokus pada tiga prinsip utama: Confidentiality (kerahasiaan), Integrity (integritas), dan Availability (ketersediaan). Ketiga prinsip ini membantu menjaga keamanan informasi dan sistem di dalam perusahaan.



Gambar 8.1 CIA Triad
https://student-activity.binus.ac.id/csc/wp-content/uploads/sites/37/2022/08/CIA-TRIAD.png

Confidentiality memastikan bahwa informasi hanya dapat diakses oleh pihak yang berwenang dan dicegah dari akses pihak yang tidak berwenang. Tujuan utamanya adalah melindungi data sensitif dari akses yang tidak sah atau kebocoran data.

Ancaman terhadap confidentiality (kerahasiaan) bertujuan untuk mendapatkan akses ilegal atau tanpa izin ke data sensitif, yang sering kali mencakup data pribadi atau data bisnis penting.

- Phishing: Serangan sosial yang mencoba untuk mencuri kredensial pengguna atau data sensitif dengan menipu pengguna agar memasukkan informasi di situs palsu. Phishing adalah ancaman besar bagi confidentiality, karena memungkinkan penyerang mendapatkan akses langsung ke informasi pribadi atau data perusahaan.

- Malware (Spyware dan Keylogger): Malware seperti spyware mengumpulkan data pengguna tanpa sepengetahuan mereka, sedangkan keylogger merekam keystrokes untuk mendapatkan informasi login atau data sensitif lainnya. Malware jenis ini dapat berisiko tinggi terhadap confidentiality dengan memberikan akses ke data pribadi atau informasi yang diinput pengguna.
- Man-in-the-Middle (MitM) Attack: Serangan ini melibatkan peretas yang memposisikan diri di antara dua pihak yang berkomunikasi, memungkinkan mereka untuk menguping atau bahkan memodifikasi data yang dikirimkan. Ini adalah ancaman besar terhadap confidentiality, terutama jika komunikasi tidak dienkripsi.
- Unauthorized Access (Akses Tidak Sah): Terjadi ketika pihak yang tidak berwenang mendapatkan akses ke sistem atau data tertentu. Ini bisa terjadi melalui kelemahan dalam sistem autentikasi atau kontrol akses, yang mengakibatkan bocornya informasi sensitif.

Implementasi atau penerapan dari kerahasiaan ini adalah sangat penting dalam menjaga privasi dan keamanan informasi sensitif. Dalam dunia yang semakin terhubung secara digital, perlindungan data pribadi dan rahasia menjadi prioritas utama untuk mencegah akses yang tidak sah dan potensi penyalahgunaan. Dengan menerapkan kebijakan kerahasiaan yang ketat, organisasi dapat memastikan bahwa hanya pihak yang berwenang yang memiliki akses ke informasi kritis, serta mematuhi peraturan dan standar privasi yang berlaku. Selain itu, edukasi dan pelatihan bagi karyawan mengenai pentingnya kerahasiaan informasi juga merupakan langkah penting dalam menciptakan budaya keamanan di lingkungan kerja.

Berikut ini beberapa contoh penerapan terhadap faktor kerahasiaan yang dapat di implementasikan pada perusahaan :

- Kerahasiaan Data Pelanggan:
    - Karyawan front desk atau admin HR hanya mengakses data pelanggan atau karyawan sesuai kebutuhan tugasnya.
    - Tindakan: Jangan berbagi password atau informasi akses kepada orang lain, bahkan rekan kerja.
    - Praktik: Gunakan password unik, aktifkan autentikasi dua faktor (2FA), dan tutup layar komputer saat meninggalkan meja.
- Mengamankan Dokumen Fisik:
    - Hindari meninggalkan dokumen penting (seperti kontrak atau formulir data pribadi) di tempat umum.
    - Tindakan: Simpan dokumen di lemari terkunci atau ruangan dengan akses terbatas.
- Berhati-hati dengan Email Phishing:
    - Jangan membuka tautan atau lampiran dari email yang tidak dikenal.
    - Praktik: Laporkan email mencurigakan ke tim IT.

Integrity memastikan bahwa data atau informasi tidak diubah oleh pihak yang tidak berwenang atau oleh kejadian yang tidak disengaja. Integritas data penting untuk memastikan keandalan dan keakuratan informasi.

Ancaman terhadap integritas berfokus pada memodifikasi data atau informasi tanpa otorisasi, sehingga menyebabkan data menjadi tidak akurat atau menyesatkan.

- Data Tampering: Perubahan data secara ilegal yang dilakukan oleh pihak tak berwenang. Contohnya adalah perubahan dalam catatan keuangan perusahaan oleh peretas, yang bisa menyebabkan kesalahan laporan atau kerugian bisnis.
- SQL Injection: Serangan ini memanfaatkan kelemahan dalam input data SQL untuk mengakses atau memodifikasi data dalam basis data secara tidak sah. SQL Injection bisa merusak integritas data dengan mengubah informasi yang tersimpan dalam basis data perusahaan.
- Replay Attack: Dalam serangan ini, peretas menangkap data yang ditransmisikan dan kemudian mengirim ulang atau memodifikasi data tersebut untuk menipu sistem atau pengguna. Misalnya, transaksi keuangan dapat direplikasi tanpa izin, yang merusak integritas sistem finansial.
- File Injection and Modification (Modifikasi File): Serangan yang dilakukan dengan memasukkan kode atau file berbahaya ke dalam sistem atau aplikasi perusahaan, yang bisa mengubah perilaku aplikasi atau mengacaukan data yang disimpan. Ancaman ini merusak integritas sistem dengan cara mengubah data atau kode asli.

Dengan menjaga integritas data, organisasi dapat membangun kepercayaan dengan pengguna dan pemangku kepentingan lainnya. Proses ini melibatkan penerapan berbagai mekanisme keamanan, seperti enkripsi, kontrol akses, dan audit rutin, untuk mendeteksi dan mencegah perubahan yang tidak sah. Selain itu, pelatihan bagi karyawan mengenai pentingnya menjaga integritas

data juga dapat berkontribusi dalam meningkatkan kesadaran dan tanggung jawab individu terhadap pengelolaan informasi. Dengan demikian, integritas data menjadi fondasi yang penting dalam pengambilan keputusan yang tepat dan efektif.

Implementasi atau penerapan dari kerahasiaan ini adalah penting untuk menjaga privasi dan keamanan informasi pribadi. Dalam dunia yang semakin terhubung secara digital, menjaga kerahasiaan data adalah prioritas utama bagi individu dan organisasi. Penerapan langkah-langkah keamanan seperti enkripsi, autentikasi dua faktor, dan kebijakan akses yang ketat dapat membantu melindungi informasi sensitif dari akses yang tidak sah. Selain itu, edukasi pengguna tentang praktik terbaik dalam menjaga privasi online juga sangat diperlukan agar mereka dapat berperan aktif dalam melindungi data mereka sendiri. Dengan demikian, kita dapat menciptakan lingkungan digital yang lebih aman dan dapat dipercaya.

Berikut beberapa contoh penerapan faktor integritas yang umum di implementasikan di dalam sebuah perusahaan:

- Menghindari Kesalahan Data:
    - Saat memasukkan data pelanggan atau transaksi, karyawan administrasi harus memverifikasi informasi sebelum menyimpannya.
    - Tindakan: Periksa data dua kali sebelum menekan tombol "simpan" atau "kirim."
- Melindungi Dokumen Digital:
    - Jangan mengedit file penting tanpa izin, terutama dokumen yang digunakan secara kolaboratif.

    - Praktik: Gunakan versi kontrol atau izin berbasis peran dalam aplikasi dokumen (seperti Google Docs atau SharePoint).

- Pencatatan dan Audit:
    - Setiap perubahan data harus dicatat dan dapat dilacak.
    - Tindakan: Masukkan komentar atau log untuk setiap perubahan dalam dokumen penting.

Availability memastikan bahwa data atau sistem tersedia untuk diakses pihak berwenang saat dibutuhkan. Prinsip dari availability ini penting untuk memastikan bisnis dapat beroperasi tanpa gangguan.

Ancaman terhadap availability (ketersediaan) bertujuan untuk mengganggu akses terhadap data atau layanan penting, membuatnya tidak dapat diakses oleh pengguna yang berwenang.

- Distributed Denial-of-Service (DDoS): Dalam serangan ini, penyerang membanjiri server atau jaringan dengan lalu lintas yang sangat besar, sehingga menyebabkan kelebihan beban dan membuat layanan tidak dapat diakses oleh pengguna yang sah. Ini adalah ancaman langsung terhadap availability, terutama pada sistem layanan publik.
- Ransomware: Jenis malware yang mengenkripsi data atau sistem perusahaan dan menahan data tersebut hingga tebusan dibayar. Ransomware adalah ancaman serius terhadap availability karena membuat data atau sistem tidak bisa diakses sampai situasi terkendali.
- Hardware Failure (Kegagalan Perangkat Keras): Meski bukan serangan langsung, kegagalan perangkat keras dapat merusak availability layanan atau data. Serangan

yang menargetkan perangkat keras, seperti overheating akibat serangan fisik atau kelalaian, juga bisa menyebabkan downtime.

- Natural Disasters (Bencana Alam): Faktor eksternal seperti gempa bumi, banjir, atau kebakaran dapat menghancurkan pusat data atau perangkat keras penting, yang menyebabkan hilangnya ketersediaan. Dalam konteks ini, disaster recovery dan backup sangat penting untuk memulihkan layanan.

Dengan adanya jaminan ketersediaan, organisasi dapat menghindari kerugian yang diakibatkan oleh downtime atau ketidakmampuan mengakses informasi penting. Strategi untuk memastikan availability meliputi penggunaan sistem cadangan, pemeliharaan rutin, dan penerapan teknologi redundansi. Hal ini juga mencakup perencanaan pemulihan bencana yang efektif untuk menangani situasi tak terduga. Dengan demikian, perusahaan dapat mempertahankan produktivitas dan menjaga kepercayaan pelanggan serta mitra bisnis.

Implementai atau penerapan faktor ketersediaan ini adalah penting untuk memastikan bahwa sumber daya yang diperlukan selalu siap ketika dibutuhkan. Dalam konteks bisnis, ini berarti memiliki stok barang yang cukup, peralatan yang dapat dioperasikan dengan baik, dan staf yang terlatih serta siap bekerja. Di sektor teknologi, ketersediaan dapat merujuk pada sistem dan jaringan yang andal, sehingga pengguna dapat mengakses layanan kapan pun diperlukan tanpa gangguan. Dengan mengelola ketersediaan secara efektif, organisasi dapat meningkatkan efisiensi operasional, meminimalkan pemborosan, dan meningkatkan kepuasan pelanggan.

Berikut ini beberapa contoh penerapan faktor ketersediaan yang umum diterapkan di dalam perusahaan :

- Jangan Matikan Peralatan Secara Mendadak:
    - Karyawan yang menggunakan aplikasi perusahaan di komputer harus memastikan perangkat dimatikan atau diperbarui sesuai arahan IT.
    - Tindakan: Ikuti prosedur resmi sebelum melakukan shutdown atau restart perangkat.
- Mendukung Backup Data:
    - Karyawan harus memahami pentingnya backup data untuk mencegah hilangnya file penting akibat kerusakan perangkat.
    - Praktik: Simpan file pekerjaan di cloud atau drive yang telah disediakan oleh perusahaan, bukan di perangkat pribadi.
- Hindari Akses Berlebih ke Sistem:
    - Jangan menggunakan perangkat perusahaan untuk hal-hal yang tidak berhubungan dengan pekerjaan, seperti mengunduh aplikasi hiburan.
    - Tindakan: Gunakan perangkat sesuai peruntukannya agar performa sistem tetap optimal.

8.2. Ancaman Siber

Ancaman siber merupakan risiko yang dihadapi oleh perusahaan dalam bentuk serangan yang dapat merusak, mencuri, atau mengakses data dan sistem tanpa izin. Ancaman ini tidak hanya berasal dari luar perusahaan, tetapi juga dapat datang dari dalam, terutama dari karyawan non IT yang mungkin tidak memiliki

pengetahuan mendalam tentang keamanan siber. Berikut adalah beberapa jenis ancaman siber yang umum terjadi di perusahaan:

- Phishing: Taktik ini melibatkan pengiriman email atau pesan yang tampak sah untuk menipu karyawan agar memberikan informasi sensitif, seperti kata sandi atau data keuangan. Karyawan non IT sering kali menjadi target karena kurangnya kesadaran tentang tanda-tanda phishing.
- Malware: Perangkat lunak berbahaya yang dirancang untuk merusak atau mengakses sistem tanpa izin. Karyawan mungkin tanpa sadar mengunduh malware melalui lampiran email atau situs web yang tidak aman.
- Insider Threats: Ancaman yang berasal dari dalam organisasi, baik secara disengaja maupun tidak. Karyawan non IT mungkin membocorkan informasi sensitif karena ketidakpahaman tentang kebijakan keamanan atau karena motivasi pribadi.
- Social Engineering: Metode manipulasi yang digunakan untuk mendapatkan informasi rahasia dengan memanipulasi karyawan. Misalnya, penyerang dapat berpura-pura sebagai IT support untuk mendapatkan akses ke sistem.
- Ransomware: Jenis malware yang mengenkripsi data perusahaan dan meminta tebusan untuk mengembalikan akses. Karyawan mungkin tidak menyadari tindakan yang dapat memicu serangan ini, seperti membuka lampiran yang mencurigakan.
- Weak Passwords: Penggunaan kata sandi yang lemah atau sama untuk berbagai akun dapat meningkatkan risiko. Karyawan non IT mungkin kurang memahami pentingnya membuat kata sandi yang kuat dan unik.

- Unpatched Software: Karyawan yang tidak melakukan pembaruan perangkat lunak secara teratur dapat membuka celah keamanan yang dapat dimanfaatkan oleh penyerang.
- Wi-Fi Tidak Aman: Risiko penyadapan data saat menggunakan Wi-Fi publik tanpa pengamanan.

Sebagai karyawan di perusahaan, Anda adalah garis pertahanan pertama dalam melindungi data perusahaan. Hal yang Bisa Anda Lakukan:

- Berhati-hati dengan Email: Periksa pengirim, jangan klik tautan mencurigakan, dan hindari lampiran yang tidak dikenal.
- Jaga Kerahasiaan Password: Gunakan password yang kuat dan autentikasi dua faktor (2FA).
- Laporkan Hal Mencurigakan: Segera beri tahu tim IT jika Anda menemukan aktivitas yang tidak biasa.
- Hindari Perangkat Tidak Dikenal: Jangan colokkan USB atau perangkat eksternal yang tidak dikenal ke komputer perusahaan.
- Verifikasi Sumber Informasi: Jangan pernah memberikan informasi sensitif kepada pihak yang tidak dikenal tanpa konfirmasi.
- Amankan Perangkat Kerja: Pastikan perangkat selalu terkunci saat tidak digunakan dan gunakan koneksi yang aman.
- Pelajari Kebijakan Perusahaan: Patuhi panduan dan kebijakan keamanan siber yang diterapkan oleh perusahaan.

Selain itu, Pastikan untuk selalu memperbarui perangkat lunak dan sistem keamanan Anda secara berkala. Ini dapat membantu melindungi dari kerentanan yang bisa dimanfaatkan oleh pihak yang tidak bertanggung jawab. Dan ikuti pelatihan keamanan yang disediakan perusahaan untuk meningkatkan pengetahuan dan kesiapsiagaan Anda terhadap ancaman siber. Dengan peran aktif dan kewaspadaan kita semua, kita dapat menjaga keamanan data perusahaan dan mendukung lingkungan kerja yang aman dan produktif.

8.3. Dampak Dari Ancaman Siber

Dampak ancaman siber pada karyawan beragam dan memengaruhi aspek kehidupan profesional serta pribadi. Berikut adalah beberapa dampaknya:

- Stres dan Kecemasan: Target serangan siber dapat mengalami stres tinggi terkait data pribadi dan profesional.
- Kehilangan Data dan Produktivitas: Serangan dapat mengakibatkan hilangnya data penting, mengganggu produktivitas.
- Pelanggaran Privasi: Risiko pencurian data pribadi dapat berdampak pada kehidupan profesional dan pribadi.
- Kerugian Finansial: Serangan siber dapat menyebabkan kerugian finansial bagi perusahaan dan karyawan.
- Dampak Reputasi: Karyawan terlibat dalam insiden siber dapat mengalami penurunan reputasi.
- Ketidakpastian Kerja: Karyawan merasa tidak aman mengenai posisi mereka setelah serangan.
- Perubahan dalam Lingkungan Kerja: Kebijakan keamanan yang lebih ketat dapat mengganggu rutinitas kerja.

- Pelatihan dan Kesadaran: Karyawan mungkin harus mengikuti pelatihan tambahan tentang keamanan siber.
- Dampak Kesehatan Mental: Ancaman siber dapat menyebabkan masalah kesehatan mental, seperti depresi.
- Meningkatnya Kerentanan: Kurangnya edukasi tentang ancaman siber membuat karyawan lebih rentan.

Dampak ancaman siber tidak hanya teknis atau finansial, tetapi juga psikologis dan sosial, memengaruhi kesejahteraan dan kinerja karyawan. Penting bagi organisasi untuk berinvestasi dalam teknologi keamanan dan pendidikan karyawan.

Dampak ancaman siber pada perusahaan sangat signifikan dan meliputi:

- Kerugian Finansial: Biaya pemulihan dan tebusan dapat mencapai jutaan.
- Kehilangan Data: Data penting seperti informasi pelanggan dan rahasia dagang dapat hilang.
- Kerusakan Reputasi: Citra perusahaan dapat rusak, mengurangi kepercayaan pelanggan.
- Gangguan Operasional: Serangan dapat menghentikan proses bisnis, mengurangi produktivitas.
- Tindakan Hukum dan Regulasi: Kebocoran data dapat mengakibatkan tuntutan hukum dan denda.
- Biaya Asuransi yang Meningkat: Premi asuransi dapat naik karena risiko yang lebih tinggi.
- Kehilangan Akses ke Sistem: Serangan ransomware dapat menghalangi akses ke data penting.

- Tantangan dalam Rekrutmen dan Retensi Karyawan: Lingkungan kerja yang tidak aman menyulitkan rekrutmen.
- Peningkatan Pengeluaran untuk Keamanan: Investasi tambahan diperlukan untuk meningkatkan keamanan siber.
- Dampak Psikologis pada Karyawan: Stres dan kecemasan di kalangan karyawan dapat meningkat.

Mengatasi ancaman siber memerlukan pendekatan proaktif dan pendidikan karyawan untuk mengurangi dampak.

CHAPTER 9
PASSWORD

Materi:

- Karakteristik password yang kuat
- Teknik menjaga keamanan password (password manager, 2FA)
- Bahaya penggunaan password yang sama

Durasi:

- 1 Jam

Tujuan:

- Memberikan pemahaman tentang bagaimana peran password dan password seperti apa yang aman.

Target Pembaca:

- Masyarakat, Mahasiswa & Karyawan Non-IT

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

9.1. Pengenalan Password

Password adalah serangkaian karakter yang digunakan untuk mengautentikasi identitas pengguna saat mengakses sistem, aplikasi, atau data tertentu. Password berfungsi sebagai lapisan perlindungan untuk menjaga keamanan informasi dan mencegah akses yang tidak sah.

Pentingnya password bagi karyawan sangatlah besar, antara lain:

- Perlindungan Data Sensitif: Password melindungi data penting perusahaan dari akses yang tidak sah. Hal ini sangat penting untuk menjaga kerahasiaan informasi pelanggan, data keuangan, dan dokumen perusahaan.
- Mencegah Pembobolan Akun: Dengan password yang kuat, risiko pembobolan akun atau penyalahgunaan akses oleh pihak yang tidak bertanggung jawab dapat diminimalkan.
- Kepatuhan terhadap Kebijakan Keamanan: Banyak organisasi memiliki kebijakan keamanan yang mengharuskan penggunaan password yang kuat dan unik. Mematuhi kebijakan ini adalah bagian dari tanggung jawab karyawan untuk menjaga keamanan data.
- Membangun Kepercayaan: Dengan menjaga keamanan informasi dan data, karyawan membantu membangun kepercayaan antara perusahaan dan klien, yang sangat penting untuk reputasi bisnis.
- Pencegahan Kerugian Finansial: Kebocoran data akibat penggunaan password yang lemah dapat menyebabkan kerugian finansial yang signifikan bagi perusahaan. Dengan memiliki password yang kuat, karyawan berkontribusi dalam mencegah kerugian tersebut.

Dalam konteks keamanan siber, penggunaan password yang baik dan kuat sangat penting untuk melindungi aset digital perusahaan dan menjaga integritas informasi yang dimiliki.

9.2. Karakter Password Yang Kuat

Password yang kuat harus:

- Minimal 12 karakter dengan kombinasi huruf besar, huruf kecil, angka, dan simbol.
- Tidak menggunakan informasi pribadi seperti nama, tanggal lahir, atau nama anggota keluarga.
- Bukan kata umum atau frasa sederhana seperti "password123" atau "qwerty".

Contoh Password Kuat:

P@ssW0rD!X23 atau MyFav0r1teP@ssw0rd!

Akan lebih sulit untuk diretas dan memberikan perlindungan tambahan terhadap informasi pribadi dan akun online Anda. Misalnya, Anda bisa membuat password dengan mengombinasikan huruf besar dan kecil, angka, dan simbol. Menggunakan kata-kata atau frasa yang memiliki makna khusus bagi Anda, tetapi tidak mudah ditebak oleh orang lain, juga bisa menjadi strategi yang efektif. Ingatlah untuk menghindari penggunaan informasi pribadi yang mudah ditemukan seperti tanggal lahir atau nama hewan peliharaan. Dengan begitu, password Anda akan lebih aman dan memberikan perlindungan yang lebih baik terhadap ancaman keamanan siber.

Berikut adalah 5 contoh password yang mengandung kekhususan dan memenuhi kriteria keamanan:

- S3mangka!Manis#2024

- KucingB3sar@Cinta
- P3rtandingan#Futbol!2024
- BukuS3jarah$P3muda!
- Cinta@Keluarga123

Pastikan untuk tidak menggunakan password yang sama di berbagai akun dan selalu perbarui secara berkala untuk menjaga keamanan.

9.3. 3 Prinsip “What”

Berikut adalah 3 cara membuat password yang kuat sesuai dengan prinsip "What":

1. What You Know:

- Gunakan kombinasi karakter: Gabungkan huruf besar, huruf kecil, angka, dan simbol untuk membuat password yang lebih sulit dipecahkan.
- Hindari informasi pribadi: Jangan gunakan informasi pribadi seperti nama, tanggal lahir, atau alamat sebagai bagian dari password Anda.
- Buat password yang unik: Gunakan password yang berbeda untuk setiap akun online.

2. What You Have:

- Gunakan otentikasi dua faktor: Tambahkan lapisan keamanan ekstra dengan menggunakan otentikasi dua faktor, seperti kode verifikasi yang dikirim melalui SMS atau aplikasi.

- Gunakan perangkat keamanan: Gunakan perangkat keamanan seperti token keamanan atau kunci USB untuk menambah perlindungan pada akun Anda.

3. What You Are:

- Gunakan biometrik: Manfaatkan fitur biometrik seperti sidik jari, pengenalan wajah, atau pemindaian iris untuk membuka kunci perangkat atau akun Anda.
- Jaga keamanan perangkat: Pastikan perangkat Anda selalu diperbarui dengan patch keamanan terbaru dan hindari menginstal aplikasi dari sumber yang tidak terpercaya.

Gambar 9.1 Jenis-Jenis Autentikasi
(Sumber: www.google.com)

Dengan mengikuti ketiga prinsip ini, Anda dapat membuat password yang lebih kuat dan aman.

9.4. Praktik Terbaik Pengelolaan Password

Pengelolaan password yang baik sangat penting untuk menjaga keamanan data dan informasi pribadi. Berikut adalah beberapa praktik terbaik dalam pengelolaan password:

- Gunakan Password yang Kuat: Buat password yang terdiri dari kombinasi huruf besar, huruf kecil, angka, dan simbol. Hindari menggunakan kata-kata umum atau informasi pribadi yang mudah ditebak.

- Buat Password yang Berbeda untuk Setiap Akun: Jangan menggunakan password yang sama untuk beberapa akun. Jika satu akun diretas, akun lain yang menggunakan password yang sama juga akan berisiko.
- Perbarui Password Secara Berkala: Ubah password Anda secara berkala, terutama untuk akun yang memiliki akses ke informasi sensitif.
- Gunakan Pengelola Password: Pertimbangkan untuk menggunakan pengelola password untuk menyimpan dan mengelola password Anda. Ini membantu Anda mengingat berbagai password yang kompleks dan unik.
- Aktifkan Autentikasi Dua Faktor (2FA): Jika memungkinkan, aktifkan 2FA untuk menambah lapisan keamanan ekstra. Ini biasanya melibatkan verifikasi identitas melalui SMS, email, atau aplikasi autentikasi.
- Hindari Menyimpan Password di Browser: Meskipun banyak browser menawarkan opsi untuk menyimpan password, ini dapat menjadi risiko keamanan. Sebaiknya gunakan pengelola password atau catat dengan aman.
- Jangan Bagikan Password: Hindari memberikan password kepada orang lain, bahkan teman dekat atau anggota keluarga, kecuali benar-benar diperlukan.
- Waspadai Phishing: Hati-hati terhadap email atau pesan yang mencurigakan yang meminta Anda untuk memberikan informasi login. Pastikan Anda hanya memasukkan password di situs yang sah dan terpercaya.

Dengan menerapkan praktik-praktik ini, Anda dapat meningkatkan keamanan akun dan melindungi informasi pribadi Anda dari ancaman yang ada.

9.4.1. Password Manager

Penggunaan password manager penting untuk menjaga keamanan informasi pribadi dan data sensitif. Dengan alat ini, Anda dapat membuat dan menyimpan kata sandi yang kuat dan unik untuk setiap akun tanpa harus mengingat semuanya. Password manager juga memudahkan pengisian otomatis informasi login, menghemat waktu, dan mengurangi risiko kesalahan. Beberapa menawarkan fitur tambahan seperti verifikasi dua langkah dan pemantauan pelanggaran data, memberikan perlindungan ekstra. Menggunakan teknologi ini adalah langkah bijak menghadapi ancaman keamanan digital yang kompleks.

Contoh tools untuk password manager :

- Bitwarden
- LastPass
- NordPass
- 1Password

9.4.2. Otentikasi Dua Faktor (2FA)

Otentikasi dua faktor sangat membantu dalam meningkatkan keamanan akun pengguna. Dengan menggabungkan sesuatu yang pengguna tahu (seperti kata sandi) dengan sesuatu yang mereka miliki (seperti ponsel untuk menerima kode verifikasi), lapisan perlindungan tambahan ini membuat lebih sulit bagi pihak yang tidak berwenang untuk mengakses informasi pribadi. Selain itu, autentikasi dua faktor juga memberikan rasa tenang bagi pengguna, mengetahui bahwa akun mereka memiliki perlindungan ekstra dari ancaman siber. Di dunia digital yang semakin kompleks dan berisiko, langkah-langkah keamanan seperti ini menjadi semakin penting untuk menjaga privasi dan integritas data.

Selalu ingat dengan 3 W yaitu :

- What You Know
- What You Have
- What You Are

CHAPTER 10
PHISHING DAN SOCIAL ENGINEERING

Materi:

- Apa itu phishing?
- Tanda-tanda email atau pesan phishing
- Contoh nyata kasus social engineering dan cara melindungi diri

Durasi:

- 1,5 Jam

Tujuan:

- Mengajarkan bagaimana mengenali dan menghindari serangan phishing serta manipulasi sosial.

Target Pembaca:

- Masyarakat, Mahasiswa & Karyawan Non-IT

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

10.1. Pengenalan Phishing

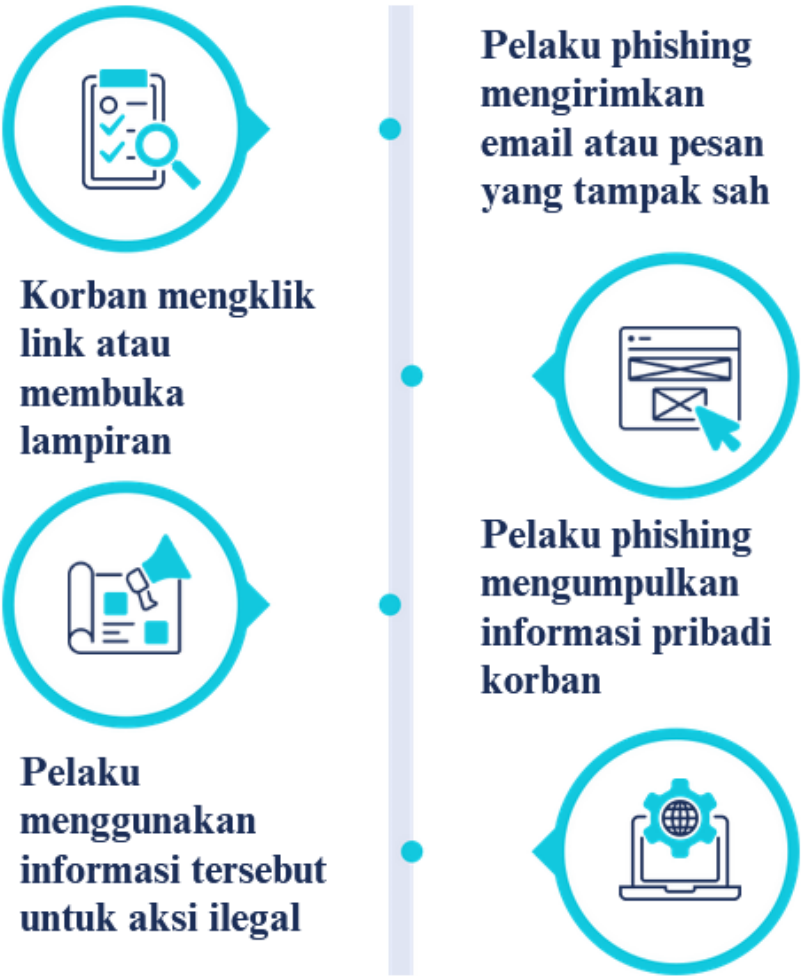
Phishing adalah upaya penipuan yang dilakukan secara online dengan tujuan untuk mencuri informasi sensitif seperti kata sandi, nomor kartu kredit, atau data pribadi lainnya. Pelaku phishing biasanya akan menyamar sebagai entitas yang terpercaya, seperti bank, perusahaan, atau bahkan teman atau keluarga. Mereka menggunakan berbagai metode untuk menipu korban, termasuk email palsu, situs web tiruan, dan pesan instan yang menarik perhatian. Penting bagi pengguna internet untuk selalu waspada dan berhati-hati saat menerima komunikasi yang mencurigakan. Menghindari mengklik tautan dari sumber yang tidak terpercaya dan memeriksa URL dengan cermat sebelum memasukkan informasi pribadi adalah langkah-langkah pencegahan yang efektif.

Phishing bukan hanya lelucon. Ini adalah kejahatan serius yang bertujuan untuk mencuri informasi pribadi Anda dan dapat menyebabkan kerugian finansial yang besar.

Email phishing adalah salah satu metode penipuan online yang paling umum. Pelaku phishing mencoba menipu korban untuk memberikan informasi pribadi atau keuangan mereka dengan menyamar sebagai entitas yang terpercaya. Berikut adalah beberapa tanda-tanda yang dapat membantu Anda mengenali email phishing:

- Alamat Email Pengirim yang Tidak Sesuai
- Permintaan Informasi Pribadi yang Tidak Biasa
- Link yang Mencurigakan
- Kesalahan Tata Bahasa atau Ejaan
- Permintaan untuk Mengklik Link atau Unduh Lampiran

Phishing adalah metode penipuan untuk mendapatkan informasi pribadi korban, seperti kata sandi dan nomor kartu kredit, dengan menyamar sebagai entitas tepercaya. Penipu menggunakan email, pesan teks, atau situs web palsu untuk menciptakan rasa panik. Untuk melindungi diri, periksa alamat email pengirim, hindari tautan mencurigakan, dan pastikan situs web aman (dengan "https"). Menggunakan perangkat lunak keamanan yang diperbarui dan autentikasi dua faktor juga meningkatkan perlindungan.



Gambar 10.1 Cara Kerja Phishing
(Sumber: Penulis)

Cara untuk mencegah serangan phishing ini sangat kompleks, karena tergantung dari psikologi manusia itu sendiri. Tetapi bukan sesuatu yang mustasil untuk mencegah serangan phishing. Berikut ini beberapa cara bagaimana kita bisa mencegah serangan phishing:

- Jangan klik link atau membuka lampiran dari email atau pesan yang tidak dikenal.
- Periksa alamat email pengirim. Email phishing seringkali berasal dari alamat email yang tidak resmi atau memiliki kesalahan ketik.
- Perhatikan tanda-tanda phishing, seperti permintaan informasi pribadi yang tidak biasa atau kesalahan tata bahasa.
- Gunakan antivirus dan firewall yang diperbarui.
- Hindari mengunduh file dari sumber yang tidak terpercaya.
- Gunakan kata sandi yang kuat dan unik untuk setiap akun online.
- Aktifkan verifikasi dua faktor untuk akun-akun penting.

Tidak peduli seberapa meyakinkan email itu terlihat, jangan pernah memberikan informasi pribadi seperti kata sandi, nomor kartu kredit, atau nomor jaminan sosial melalui email.

10.2. Pengenalan Social Engineering

Social engineering adalah teknik manipulasi psikologis yang digunakan untuk mendapatkan informasi sensitif atau akses ke sistem komputer dari individu atau organisasi. Pelaku social engineering memanfaatkan kelemahan manusia, seperti rasa kepercayaan, keserakahan, atau rasa takut, untuk memanipulasi korban agar melakukan tindakan yang diinginkan. Dengan berbagai metode, pelaku sering berpura-pura menjadi pihak yang dapat dipercaya, seperti petugas keamanan atau rekan kerja, untuk memperoleh informasi seperti kata sandi atau detail pribadi. Salah satu bentuk umum social engineering adalah phishing, di mana korban diarahkan untuk mengunjungi situs web palsu dan memasukkan informasi mereka.

Social engineering dapat memiliki konsekuensi yang serius bagi individu dan organisasi. Beberapa bahaya utama dari social engineering meliputi:

- Pencurian identitas yang mengakibatkan kerugian finansial dan reputasi.
- Kerugian finansial besar akibat penyalahgunaan informasi keuangan.
- Kerusakan reputasi jika informasi pribadi dipublikasikan tanpa izin.
- Akses tidak sah ke sistem komputer organisasi.
- Kerugian bisnis signifikan bagi organisasi, termasuk kehilangan pelanggan dan penurunan pendapatan.

Teknik Rekayasa Sosial (Social Engineering) meliputi:

- Phishing: Mengelabui korban dengan pesan dari sumber terpercaya untuk memberikan informasi sensitif.

- Quid Pro Quo: Menawarkan imbalan untuk mendapatkan informasi sensitif, seperti hadiah untuk mengisi survei.
- Baiting: Menarik korban dengan tawaran menarik seperti perangkat USB terinfeksi.
- Pretexting: Menciptakan alasan meyakinkan untuk menghubungi korban, seperti mengaku sebagai petugas bank.
- Spear Phishing: Phishing yang lebih terarah, menargetkan individu tertentu dengan informasi spesifik.

Manipulasi Emosi:

- Fear: Menciptakan rasa takut untuk mendorong tindakan cepat, seperti ancaman pemblokiran akun.
- Greed: Menawarkan hadiah besar untuk menarik perhatian korban.
- Curiosity: Membangkitkan rasa ingin tahu dengan informasi samar.

Tekanan Sosial:

- Authority: Pelaku berpura-pura sebagai sosok berwenang untuk memberi perintah.
- Consensus: Pelaku menciptakan kesan bahwa banyak orang melakukan sesuatu agar korban ikut.
- Liking: Pelaku membangun hubungan baik dengan korban sebelum meminta sesuatu.

Teknik Lain:

- Tailgating: Mengikuti orang dengan akses tanpa izin.

- Dumpster diving: Mencari informasi sensitif di tempat sampah.
- Shoulder surfing: Mengamati korban saat mengetik password atau PIN.

Pelaku social engineering bekerja dengan cerdik, mereka memulai dengan mengumpulkan informasi sebanyak mungkin tentang targetnya. Kemudian, mereka membangun hubungan kepercayaan dengan korban, seringkali dengan menyamar sebagai orang yang dikenal atau pihak berwenang. Setelah itu, mereka akan memanfaatkan kelemahan psikologis manusia seperti rasa takut, keserakahan, atau rasa ingin tahu untuk memanipulasi korban.

Gambar 10.2 Cara Kerja Socia Engineering
(Sumber: Penulis)

Beberapa cara yang dapat diterapkan agar terhindar dari serangan social engineering adalah sebagai berikut:

- Tingkatkan Kesadaran
- indungi Informasi Pribadi
- Verifikasi Sumber
- Berhati-hati dengan Teknik Manipulasi
- Praktik Kerja yang Aman
- Budaya Keamanan

Pelaku social engineering terus mengembangkan taktik mereka, sehingga kita harus selalu waspada dan tidak mudah percaya pada informasi yang kita terima, terutama jika melibatkan permintaan informasi pribadi atau keuangan.

10.3. Contoh Kasus Pada Perusahaan

Phishing Melalui Email di Perusahaan Teknologi

Deskripsi: Sebuah perusahaan teknologi sedang dalam proses meluncurkan aplikasi baru. Karyawan menerima email yang tampaknya berasal dari departemen IT yang meminta mereka untuk memperbarui kredensial login mereka untuk mengakses sistem internal. Email tersebut mencantumkan tautan untuk masuk, yang sebenarnya adalah situs web palsu yang dirancang untuk mencuri informasi kredensial pengguna.

Dampak: Beberapa karyawan yang tidak curiga mengklik tautan dan memasukkan informasi login mereka. Akibatnya, penyerang dapat mengakses akun internal mereka, yang kemudian digunakan untuk mendapatkan akses ke data sensitif perusahaan, termasuk informasi aplikasi baru yang belum dirilis. Kerugian ini mencakup potensi kebocoran data dan merusak kepercayaan pelanggan.

Pencegahan:

- Pelaksanaan pelatihan keamanan siber untuk semua karyawan tentang tanda-tanda phishing.
- Penggunaan filter email yang lebih baik untuk mendeteksi email yang mencurigakan.
- Implementasi otentikasi dua faktor untuk memberikan lapisan keamanan tambahan.

Phishing Melalui Pesan Teks (Smishing) di Perusahaan Ritel

Deskripsi: Perusahaan ritel besar menerima laporan dari staf toko bahwa mereka mendapatkan pesan SMS yang mengaku berasal dari manajer area yang meminta mereka untuk mengonfirmasi informasi akun dengan mengklik tautan yang disertakan. Pesan tersebut menyebutkan bahwa akan ada audit mendadak dan karyawan harus memberikan informasi segera.

Dampak: Beberapa karyawan mengklik tautan dan mengisinya dengan informasi pribadi, termasuk nomor jaminan sosial dan rincian rekening bank. Data ini kemudian digunakan oleh penyerang untuk melakukan penipuan identitas, yang menyebabkan kerugian finansial bagi karyawan dan perusahaan dalam kerugian reputasi.

Pencegahan:

- Penyuluhan karyawan tentang praktik keamanan yang baik, termasuk cara mengenali smishing.
- Memastikan semua komunikasi penting dilakukan melalui saluran resmi yang telah terverifikasi.

Pretexting dalam Perusahaan Keuangan

Deskripsi: Seorang penyerang berpura-pura menjadi perwakilan dari lembaga pengatur keuangan. Dalam panggilan telepon ke departemen keuangan perusahaan, penyerang meminta akses untuk informasi sensitif terkait laporan keuangan untuk keperluan audit. Korban, yang tidak menaruh curiga, memberikan informasi tanpa memverifikasi identitas penelepon.

Dampak: Penyerang berhasil mengumpulkan data keuangan yang sensitif, yang kemudian digunakan untuk melakukan pencurian identitas atau penipuan terhadap perusahaan. Hal ini menciptakan risiko besar bagi perusahaan, termasuk potensi sanksi dari regulator dan kerugian finansial yang signifikan.

Pencegahan:

- Penguatan kebijakan verifikasi identitas untuk semua komunikasi terkait informasi sensitif.
- Pelatihan umum untuk semua karyawan mengenai teknik social engineering dan cara menanggapi permintaan informasi yang tidak terverifikasi.

Baiting di Perusahaan Produksi

Deskripsi: Seorang penyerang meninggalkan USB flash drive di area parkir perusahaan produksi dengan label menarik yang bertuliskan "Bonus Karyawan 2024." Seorang karyawan yang penasaran mengambil flash drive tersebut dan memasukkannya ke dalam komputer perusahaan untuk melihat informasi lebih lanjut tentang bonus.

Dampak: Flash drive tersebut berisi malware yang secara otomatis menginfeksi sistem ketika dipasang. Penyerang mendapatkan

akses ke jaringan internal perusahaan, yang memungkinkan mereka untuk mencuri data sensitif dan melakukan serangan lebih lanjut.

Pencegahan:

- Pendidikan kepada karyawan tentang bahaya menggunakan perangkat eksternal yang tidak dikenal.
- Implementasi kebijakan untuk tidak mengizinkan penggunaan perangkat penyimpanan luar tanpa verifikasi.

Kasus-kasus phishing dan social engineering di atas menunjukkan berbagai metode yang digunakan oleh penyerang untuk menargetkan perusahaan dan karyawan mereka. Keberhasilan serangan ini sering kali bergantung pada kurangnya kesadaran keamanan di kalangan karyawan. Oleh karena itu, pelatihan yang baik dan strategi pencegahan yang efektif sangat penting untuk melindungi organisasi dari ancaman ini. Dengan meningkatkan kesadaran dan pelatihan tentang praktik keamanan terbaik, perusahaan dapat mengurangi risiko serangan yang berhasil.

CHAPTER 11
KEAMANAN DI MEDIA SOSIAL DAN EMAIL

Materi:

- Privasi di media sosial
- Tips menggunakan email secara aman
- Menghindari tautan berbahaya

Durasi:

- 1 Jam

Tujuan:

- Mengetahui cara menjaga keamanan informasi pribadi di media sosial dan email.

Target Pembaca:

- Masyarakat, Mahasiswa & Karyawan Non-IT

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

11.1. Privasi di Media Sosial

Privasi di media sosial mengacu pada kontrol atas informasi pribadi yang dibagikan secara online. Meskipun media sosial menghubungkan kita, penting untuk berhati-hati karena informasi dapat diakses oleh banyak orang. Langkah-langkah untuk melindungi privasi termasuk mengatur pengaturan privasi, menggunakan kata sandi yang kuat, dan hanya menerima permintaan pertemanan dari orang yang dikenal. Kita juga harus waspada terhadap penipuan online dan phishing, serta memverifikasi informasi sebelum membagikannya. Dengan menjaga privasi, kita dapat menikmati manfaat teknologi tanpa mengorbankan keamanan pribadi.

Mengapa Privasi di Media Sosial Penting?

- Perlindungan Identitas: Informasi pribadi yang dibagikan secara terbuka dapat disalahgunakan oleh pihak yang tidak bertanggung jawab untuk melakukan penipuan identitas.
- Pelindungan Reputasi: Postingan yang tidak bijaksana atau informasi pribadi yang bocor dapat merusak reputasi kita baik di dunia online maupun di kehidupan nyata.
- Pencegahan Spam: Semakin banyak informasi pribadi yang kita bagikan, semakin besar kemungkinan kita akan menjadi target spam dan iklan yang tidak relevan.
- Keamanan Data: Data pribadi yang kita bagikan di media sosial dapat menjadi sasaran peretas.

11.1.1. Jenis-Jenis Data Sensitif

Data sensitif adalah jenis informasi yang jika bocor dapat menyebabkan kerugian signifikan bagi individu atau organisasi. Di era digital, terutama di platform media sosial, data sensitif menjadi komoditas yang sangat berharga bagi para pelaku

kejahatan siber. Berikut adalah beberapa contoh data sensitif yang sering menjadi target:

Tabel 11.1 Jenis-Jenis Data Sensitif

Interaksi sosial	Riwayat pesan	Riwayat pembelian
Foto dan video	Posting dan komentar	Data pribadi
Data keuangan	Data kesehatan	Data lokasi
Riwayat penelusuran	Preferensi Pribadi	Kontak Sosial

Data Pribadi Identifikasi

- Informasi Identitas Lengkap: Nama lengkap, alamat lengkap, nomor telepon, tanggal lahir, nomor KTP, nomor paspor.
- Biodata Keluarga: Informasi tentang keluarga, termasuk nama anggota keluarga, hubungan kekerabatan, dan tanggal lahir.
- Informasi Keuangan: Nomor rekening bank, nomor kartu kredit, informasi pembayaran online, riwayat transaksi.
- Data Biometrik: Sidik jari, wajah, iris mata, dan data biometrik lainnya yang digunakan untuk identifikasi.

Data Pribadi Sensitif

- Data Kesehatan: Riwayat penyakit, hasil tes medis, catatan medis, informasi genetik.
- Data Seksual: Orientasi seksual, identitas gender, riwayat hubungan seksual.

- Data Politik: Opini politik, afiliasi partai politik, aktivitas politik.
- Data Agama: Keyakinan agama, praktik keagamaan.

Data Lokasi

- Riwayat Lokasi: Data GPS yang menunjukkan lokasi pengguna secara real-time atau historis.
- Tempat yang Sering Dikunjungi: Tempat-tempat yang sering dikunjungi oleh pengguna, seperti rumah, kantor, sekolah, dan tempat umum lainnya.

Data Perilaku

- Preferensi Pribadi: Hobi, minat, kebiasaan, dan preferensi lainnya.
- Kontak Sosial: Daftar teman, keluarga, dan kolega di media sosial.
- Interaksi Online: Riwayat pencarian, postingan, komentar, dan pesan pribadi.

11.1.2. Risiko Kebocoran Data Sensitif

Kebocoran data sensitif dapat mengakibatkan berbagai konsekuensi negatif, antara lain:

- Pencurian Identitas: Pelaku kejahatan dapat menggunakan data pribadi Anda untuk membuka rekening bank baru, melakukan penipuan, atau mengambil alih akun online Anda.
- Pekerasan: Informasi pribadi yang sensitif, seperti alamat rumah atau tempat kerja, dapat digunakan untuk melakukan tindakan kekerasan atau ancaman.

- Pencemaran Nama Baik: Informasi pribadi yang bersifat pribadi atau memalukan dapat disebarluaskan secara online, merusak reputasi Anda.
- Diskriminasi: Informasi sensitif seperti ras, agama, atau orientasi seksual dapat digunakan untuk mendiskriminasi Anda dalam pekerjaan, pendidikan, atau kehidupan sosial.
- Kerugian Finansial: Kebocoran data keuangan dapat mengakibatkan kerugian finansial yang besar.

Oleh karena itu, perlu hindari berbagi data sensitif terutama di media sosial. Sebelum memposting sesuai, pikirkan terlebih dahulu apakah data atau informasi ini sensitif atau tidak. Berikut Contoh Data yang Tidak Boleh Dibagikan:

- Foto KTP atau Paspor: Dokumen ini mengandung banyak informasi pribadi yang sangat sensitif.
- Detail Transaksi: Hindari membagikan bukti pembayaran atau screenshot transaksi online.
- Status Hubungan: Informasi tentang hubungan pribadi bisa menjadi sasaran untuk penipuan atau gangguan.
- Rencana Liburan: Mengumumkan rencana liburan secara detail dapat mengundang pencuri.
- Informasi Kontak Anak: Hindari membagikan informasi kontak anak-anak, seperti nomor telepon atau sekolah.
- Foto yang Provokatif: Foto yang terlalu terbuka atau yang dapat ditafsirkan secara berbeda oleh orang lain.
- Opini Politik atau Agama yang Ekstrem: Postingan yang terlalu ekstrem dapat mengundang perdebatan yang tidak perlu.

11.2. Keamanan Penggunaan Email

Email telah menjadi alat komunikasi utama dalam kehidupan pribadi dan profesional. Namun, seiring dengan kemudahannya, email juga rentan terhadap berbagai ancaman keamanan. Oleh karena itu, penting untuk memahami dan menerapkan langkah-langkah keamanan yang tepat untuk melindungi komunikasi melalui email.

Mengamankan penggunaan email itu sangat penting. Berikut beberapa alasannya :

- Kerahasiaan: Email seringkali berisi informasi sensitif seperti data pribadi, informasi keuangan, atau rahasia bisnis. Jika email diretas, informasi tersebut dapat disalahgunakan oleh pihak yang tidak bertanggung jawab.
- Integritas: Pesan email dapat diubah atau dimanipulasi selama proses pengiriman. Hal ini dapat menyebabkan kesalahpahaman, kerugian finansial, atau bahkan masalah hukum.
- Ketersediaan: Layanan email dapat mengalami gangguan atau serangan yang menyebabkan pengguna tidak dapat mengakses email mereka.

11.2.1. Verifikasi alamat pengirim

Memverifikasi alamat pengirim email sangat penting untuk memastikan bahwa pesan yang Anda terima benar-benar berasal dari pengirim yang tertera. Ini menjadi semakin krusial di era di mana serangan phishing dan spam semakin marak. Berikut beberapa tip untuk melakukan verifikasi pengirim email:

- Periksa Alamat Email
- Tinjau Isi Pesan
- Gunakan Fitur Verifikasi Email

- Hubungi Pengirim Secara Langsung
- Waspadai Email yang Terlalu Baik untuk Menjadi Benar

11.2.2. Menghindari lampiran berbahaya

Lampiran email seringkali menjadi pintu masuk bagi malware dan virus untuk menginfeksi perangkat Anda. Berikut beberapa tips untuk menjaga keamanan Anda:

- Jangan Buka Lampiran dari Pengirim yang Tidak Dikenal:
    - Hati-hati dengan Email Phishing: Email phishing seringkali menyertakan lampiran berbahaya dengan iming-iming menarik, seperti undian atau penawaran pekerjaan.
    - Verifikasi Pengirim: Pastikan Anda benar-benar mengenal pengirim sebelum membuka lampiran.
- Periksa Ekstensi File:
    - Waspadai Ekstensi Berbahaya: Ekstensi file seperti .exe, .scr, .com, .bat, dan .vbs seringkali digunakan untuk menyembunyikan malware.
    - Ekstensi yang Disamarkan: Terkadang, ekstensi file asli disamarkan dengan mengubah nama file. Jadi, jangan hanya melihat nama file, tetapi juga perhatikan ekstensinya.
- Gunakan Perangkat Lunak Antivirus:
    - Pemindaian Otomatis: Pastikan perangkat lunak antivirus Anda selalu aktif dan secara otomatis memindai semua lampiran email.
    - Pembaruan Berkala: Selalu perbarui perangkat lunak antivirus Anda agar tetap efektif dalam mendeteksi ancaman terbaru.

- Hindari Mengklik Tautan dalam Email:
    - Tautan Phishing: Banyak email phishing mengandung tautan yang mengarahkan Anda ke situs web palsu yang dirancang untuk mencuri informasi pribadi Anda.
    - Verifikasi Tautan: Sebelum mengklik tautan, periksa dengan cermat URL tujuan. Pastikan itu adalah situs web yang resmi dan terpercaya.
- Jangan Menyimpan Lampiran yang Tidak Diperlukan:
    - Bersihkan Kotak Masuk: Hapus lampiran yang tidak diperlukan
    - Simpan di Tempat yang Aman: Jika Anda perlu menyimpan lampiran, simpan di folder yang terpisah.
- Aktifkan Fitur Keamanan Email:
    - Filter Spam: Gunakan fitur filter spam pada layanan email Anda untuk menyaring email yang mencurigakan.
    - Autentikasi Dua Faktor: Aktifkan autentikasi dua faktor untuk melindungi akun email Anda dari akses yang tidak sah.
- Tingkatkan Kesadaran:
    - Tetap Waspada: Selalu waspada terhadap email yang mencurigakan dan jangan mudah tertipu oleh iming-iming yang terlalu bagus.
    - Pelajari Tanda-Tanda Email Phishing: Pelajari ciri-ciri umum email phishing, seperti kesalahan tata

bahasa, permintaan informasi pribadi, atau tautan yang mencurigakan.

11.3. Contoh Kasus Media Sosial di Perusahaan

Staf Mengungkapkan Opini Negatif

Deskripsi: Seorang karyawan di perusahaan retail besar memposting di Twitter tentang pengalaman buruknya dengan sistem manajemen perusahaan. Ia mencurahkan unek-unek mengenai lingkungan kerja yang toxic dan bagaimana kebijakan perusahaan yang tidak mendukung karyawan mempengaruhi produktivitas.

Dampak: Postingan tersebut menjadi viral setelah menarik perhatian publik dan media. Reputasi perusahaan terguncang dan pelanggan mulai mempertanyakan nilai-nilai perusahaan. Hal ini juga menyebabkan beberapa karyawan lain mengikuti jejak yang sama, memperburuk citra perusahaan di mata publik.

Tindakan yang Diambil:

- Perusahaan merespons dengan cepat dengan mengeluarkan pernyataan publik yang mengakui masalah tersebut dan menjanjikan peninjauan kebijakan internal.
- Mereka juga meluncurkan kampanye untuk meningkatkan komunikasi jagung antar-departemen agar semua suara karyawan didengar.

Serangan Siber melalui Media Sosial

Deskripsi: Perusahaan teknologi mengalami serangan siber ketika penyerang membuat akun palsu yang menyamar sebagai akun resmi perusahaan di Facebook. Akun ini mulai mengirimkan

tautan phishing kepada pengikut yang mengaku sebagai penawaran eksklusif.

Dampak: Beberapa pelanggan terpancing untuk mengklik tautan tersebut, yang mengakibatkan pencurian data pribadi dan informasi kartu kredit. Perusahaan kehilangan kepercayaan dari pelanggannya dan menghadapi risiko hukum akibat kebocoran data.

Tindakan yang Diambil:

- Perusahaan segera menutup akun palsu dan melaporkan kepada platform media sosial.
- Mereka juga mempublikasikan peringatan kepada pelanggan untuk tidak mengklik tautan yang mencurigakan dan untuk selalu memverifikasi keaslian akun sebelum berinteraksi.

Krisis Reputasi karena Konten yang Tidak Peka

Deskripsi: Sebuah perusahaan fashion meluncurkan kampanye pemasaran di Instagram yang menampilkan model dalam konteks budaya tertentu tanpa menghormati nilai-nilai dan tradisi budaya tersebut. Konten tersebut dianggap ofensif, dan banyak pengguna media sosial berbondong-bondong mengecam kampanye tersebut.

Dampak: Kampanye yang awalnya dimaksudkan untuk menarik perhatian malah menimbulkan kemarahan publik, memicu #BoycottBrandX di media sosial. Perusahaan harus menghadapi protes, kritik ketidakpedulian terhadap keberagaman dan kesensitifan budaya.

Tindakan yang Diambil:

- Perusahaan mengeluarkan permintaan maaf publik dan menarik kampanye tersebut.
- Mereka juga berkomitmen untuk melakukan konsultasi dengan para ahli budaya ketika merancang konten di masa depan dan mengadopsi pendekatan yang lebih inklusif.

Pengelolaan Ulasan Negatif

Deskripsi: Sebuah restoran populer menerima ulasan negatif di platform media sosial seperti Google Reviews dan Yelp. Pengunjung menggambarkan pengalaman buruk pada pelayanan dan kualitas makanan. Ulasan ini mulai menarik perhatian banyak orang.

Dampak: Ulasan negatif tersebut secara langsung mempengaruhi jumlah pengunjung yang datang. Penurunan pendapatan yang signifikan terlihat dalam beberapa minggu setelah munculnya ulasan tersebut.

Tindakan yang Diambil:

- Restoran tersebut mengambil langkah proaktif dengan merespons setiap ulasan negatif secara terbuka, menawarkan solusi, dan meminta pengunjung untuk memberikan kesempatan kedua.
- Mereka juga melakukan perbaikan layanan dan melakukan kampanye promosi untuk menarik pelanggan kembali.

Kegiatan Karyawan di Media Sosial Mengundang Kontroversi

Deskripsi: Seorang karyawan di perusahaan konsultan memposting foto di Instagram dengan caption yang dianggap rasisme atau tidak pantas. Meskipun itu adalah pendapat pribadi, hal ini menarik perhatian publik dan media, menimbulkan kontroversi.

Dampak: Perusahaan mendapat banyak kritik karena dianggap mendukung pandangan tersebut, meskipun itu bukan representasi perusahaan. Reputasi perusahaan terancam dan mereka mengalami eksposur media yang negatif.

Tindakan yang Diambil:

- Perusahaan merilis pernyataan resmi yang menjelaskan bahwa pandangan karyawan tersebut tidak mencerminkan nilai-nilai perusahaan.
- Mereka juga melihat melalui kebijakan media sosial dan memberikan pelatihan kepada karyawan tentang etika dalam menggunakan media sosial dan dampak dari konten yang mereka bagi dengan publik.

CHAPTER 12
KEAMANAN PERANGKAT DAN JARINGAN

Materi:

- Keamanan perangkat (PC, laptop, smartphone)
- Risiko penggunaan Wi-Fi publik
- Mengapa penting untuk memperbarui perangkat lunak secara berkala

Durasi:

- 1 Jam

Tujuan:

- Memahami bagaimana menjaga perangkat dan jaringan dari ancaman keamanan.

Target Pembaca:

- Masyarakat, Mahasiswa & Karyawan Non-IT

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

12.1. Pengenalan Keamanan Perangkat

Keamanan Perangkat adalah serangkaian langkah dan tindakan yang dilakukan untuk melindungi perangkat elektronik Anda, seperti komputer, laptop, smartphone, atau tablet, dari ancaman keamanan siber. Ancaman ini bisa berupa virus, malware, peretas, atau serangan siber lainnya yang bertujuan untuk mencuri data pribadi, merusak sistem, atau mengambil alih kendali perangkat Anda.

Beberapa jenis perangkat komputasi yang umum digunakan oleh Masyarakat yang perlu di amankan adalah sebagai berikut:

- Komputer
- Laptop
- Smartphone
- Tablet
- Mac

Mengamankan perangkat anda merupakan langkah penting untuk melindungi aset dari ancaman. Beberapa alasan mengapa mengamankan perangkat itu penting adalah :

- Data-data penting seperti informasi keuangan, kontak, dan foto pribadi tersimpan di perangkat Anda.
- Malware dapat merusak sistem operasi, aplikasi, dan file-file penting di perangkat Anda.
- Informasi pribadi yang dicuri dari perangkat Anda dapat digunakan oleh penipu untuk melakukan pencurian identitas.
- Perangkat yang terinfeksi dapat menjadi pintu masuk bagi penyerang untuk menyerang jaringan yang lebih besar, seperti jaringan perusahaan.

Dengan mengambil langkah-langkah ini, Anda dapat mengurangi risiko dan melindungi perangkat serta data pribadi Anda dari ancaman yang tidak diinginkan. Keamanan perangkat bukan hanya tentang melindungi informasi pribadi, tetapi juga menjaga integritas dan fungsionalitas perangkat yang Anda

Elemen Utama Keamanan Perangkat

Keamanan perangkat melibatkan berbagai lapisan perlindungan untuk menjaga data Anda tetap aman. Berikut adalah beberapa elemen utamanya:

- Perangkat Lunak Antivirus: Melindungi perangkat dari virus, malware, dan ancaman lainnya.
- Firewall: Membentuk perisai antara perangkat Anda dan internet, memblokir lalu lintas yang mencurigakan.
- Sistem Operasi yang Terbarui: Pembaruan sistem operasi seringkali mencakup perbaikan keamanan untuk mengatasi kerentanan yang diketahui.
- Kata Sandi yang Kuat: Gunakan kata sandi yang unik dan kompleks untuk setiap akun.
- Autentikasi Dua Faktor: Tambahkan lapisan keamanan ekstra dengan menggunakan autentikasi dua faktor.
- Cadangan Data: Lakukan backup data secara teratur untuk mencegah kehilangan data jika terjadi hal yang tidak diinginkan.

Tips Tambahan

- Jangan Root atau Jailbreak Perangkat: Modifikasi sistem operasi dapat membuat perangkat Anda lebih rentan terhadap serangan.

- Periksa Aplikasi Sebelum Diinstal: Hanya instal aplikasi dari sumber yang terpercaya.
- Perhatikan Izin Aplikasi: Berikan izin akses aplikasi hanya yang benar-benar diperlukan.
- Jaga Kerahasiaan Informasi Pribadi: Jangan sembarangan membagikan informasi pribadi Anda secara online.
- Hindari Wi-Fi Publik yang Tidak Aman: Gunakan VPN saat terhubung ke jaringan Wi-Fi publik.

12.2. Bahaya dari Wi-Fi publik

Wi-Fi publik adalah jaringan nirkabel yang dapat diakses oleh siapa saja secara gratis di tempat-tempat umum seperti kafe, bandara, hotel, atau ruang publik lainnya. Jaringan ini memungkinkan pengguna untuk terhubung ke internet tanpa perlu menggunakan data seluler.

Keuntungan Wi-Fi Publik:

- Gratis: Tidak perlu mengeluarkan biaya tambahan untuk mengakses internet.
- Mudah Diakses: Tersedia di banyak tempat umum.

Meskipun mudah diakses, Wi-Fi publik memiliki beberapa risiko keamanan yang perlu diperhatikan:

- Peretas dapat dengan mudah menyadap koneksi Wi-Fi publik untuk mencuri data pribadi seperti kata sandi, nomor kartu kredit, atau informasi sensitif lainnya.
- Perangkat Anda dapat terinfeksi malware melalui Wi-Fi publik, terutama jika Anda mengunduh file atau mengunjungi situs web yang tidak aman.

- Serangan ini memungkinkan peretas untuk menyisipkan diri di antara koneksi Anda dan server yang Anda hubungi, sehingga mereka dapat mencuri data yang Anda kirim atau terima.
- Peretas dapat membuat hotspot Wi-Fi palsu dengan nama yang mirip dengan hotspot resmi untuk menjebak pengguna agar terhubung ke jaringan mereka.

Wi-Fi publik memang menawarkan kemudahan akses internet, namun pengguna perlu waspada terhadap risiko keamanan yang menyertainya. Beberapa ancaman yang dapat terjadi terkait Wi-Fi publik adalah sebagai berikut:

- Sniffing
- Malware
- Man-in-the-Middle Attack
- Evil Twin

Apa itu man-in-the-middle attack?

Serangan Man-in-the-Middle (MitM) adalah jenis serangan siber di mana penyerang diam-diam menyisipkan dirinya di antara dua pihak yang sedang berkomunikasi. Penyerang ini dapat mencegat, memodifikasi, atau bahkan mencuri data yang dikirimkan antara kedua pihak tersebut. Bayangkan seperti seseorang yang menyadap percakapan telepon Anda tanpa Anda sadari.

Ketika terhubung ke Wi-Fi publik, Anda berbagi jaringan dengan banyak orang, berisiko menjadi korban serangan MitM. Contoh serangan termasuk:

- Hotspot Palsu: Penyerang membuat jaringan Wi-Fi mirip dengan yang asli untuk mencuri data.
- Packet Sniffing: Penyerang mencegat paket data untuk melihat informasi sensitif.

- DNS Spoofing: Penyerang mengarahkan Anda ke situs web palsu untuk mencuri informasi login.

Cara Kerja Serangan MitM:

- Penyerang membuat jaringan palsu.
- Korban terhubung tanpa sadar.
- Data dilewatkan melalui perangkat penyerang.
- Penyerang mencegat dan merekam data.

Berikut contoh fake hotspot atau hotspot palsu yang dapat beredar di area publik tanpa anda sadari:

“Starbucks_FreeWiFi" yang asli vs "Starbucks_FreeWiFi_Free" yang palsu.

Mungkin bagi Masyarakat awam penamaan hotspot diatas sama saja. Sehingga mereka dapat masuk ke hotspot mana saja dan ketika mereka salah masuk hotspot, maka hamper semua aktifitas mereka di jaringan tsb atau internet akan direkam oleh penyerang. Dan hal ini tentu saja dapat mengakibatkan data kita dicuri, dll.

Gunakan VPN saat mengakses jaringan publik

Sangat disarankan untuk menggunakan VPN (Virtual Private Network) saat mengakses jaringan publik. VPN adalah layanan yang memungkinkan Anda membuat koneksi internet pribadi yang terenkripsi. Ini berarti data yang Anda kirim dan terima akan dienkripsi, sehingga sulit bagi pihak ketiga, termasuk peretas, untuk mencegat dan membaca data tersebut.

Pentingnya menggunakan VPN dalam jaringan publik adalah :

- Enkripsi Data: VPN mengenkripsi semua data yang Anda kirim dan terima, sehingga melindungi informasi sensitif

Anda seperti kata sandi, nomor kartu kredit, dan data pribadi lainnya.

- Sembunyikan Alamat IP: VPN menyembunyikan alamat IP asli Anda, sehingga sulit bagi orang lain untuk melacak aktivitas online Anda.
- Lindungi dari Serangan Man-in-the-Middle: VPN membantu mencegah serangan MitM dengan mengenkripsi lalu lintas data Anda.
- Akses Konten yang Diblokir: VPN memungkinkan Anda mengakses situs web dan layanan yang diblokir di wilayah tertentu.

Beberapa tools atau aplikasi VPN yang bisa digunakan sangat banyak, apakah anda ingin berbayar atau gratis. Berikut contoh tools VPN yang umum digunakan:

- NordVPN
- ExpressVPN
- Surfshark
- ProtonVPN

Pilihan VPN terbaik dapat bervariasi berdasarkan kebutuhan pribadi Anda, seperti prioritas keamanan, kecepatan, dan fitur-fitur tambahan. Sebelum memilih VPN, pastikan untuk melakukan riset dan membaca ulasan pengguna untuk menemukan yang paling sesuai dengan Anda.

CHAPTER 13
PROSEDUR INSIDEN KEAMANAN

Materi:

- Bagaimana melaporkan insiden keamanan
- Tindakan pertama saat terkena serangan (misalnya ransomware)
- Proses recovery dan mitigasi

Durasi:

- 1 Jam

Tujuan:

- Mengetahui langkah-langkah yang harus dilakukan jika terjadi insiden keamanan.

Target Pembaca:

- Masyarakat, Mahasiswa & Karyawan Non-IT

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

13.1. Mengenali Insiden Keamanan

Insiden keamanan adalah suatu kejadian yang mengakibatkan pelanggaran atau ancaman terhadap kerahasiaan, integritas, atau ketersediaan sistem informasi. Dengan kata lain, ini adalah peristiwa yang mengakibatkan kebocoran data, kerusakan sistem, atau gangguan layanan yang tidak diinginkan.

Beberapa bentuk insiden keamanan antara lain sebagai berikut:

- Ransomware
- Phishing
- DDoS (Distributed Denial of Service)
- Malware
- Social Engineering

Insiden keamanan dapat terjadi dalam berbagai bentuk, mulai dari serangan siber seperti peretasan, malware, dan ransomware, hingga kesalahan manusia seperti kehilangan perangkat atau kesalahan konfigurasi.

Gejala serangan ransomware

Berikut adalah beberapa gejala umum yang mengindikasikan adanya serangan ransomware:

- Pesan Tebusan: Munculnya pesan yang meminta pembayaran tebusan untuk memulihkan data. Pesan ini biasanya ditampilkan dalam bentuk pop-up, file teks, atau perubahan pada wallpaper desktop.
- File Tidak Dapat Dibuka: File-file penting tiba-tiba tidak dapat dibuka atau diakses. File-file tersebut biasanya memiliki ekstensi file yang berbeda atau ditambahkan dengan kata-kata tertentu (misalnya, .locked, .encrypted).

- Performa Sistem Melambat: Komputer atau jaringan menjadi lambat atau tidak responsif.
- Proses yang Tidak Dikenal: Munculnya proses atau program yang tidak dikenal berjalan di latar belakang.
- Aktivitas Jaringan yang Tidak Biasa: Meningkatnya aktivitas jaringan yang tidak biasa, seperti transfer data dalam jumlah besar atau koneksi ke server yang tidak dikenal.

Jenis-Jenis ransomware yang umum ditemukan di internet dan terjadi atau menyerang perusahaan adalah:

- Locker ransomware: Mengunci perangkat korban, mencegah akses ke sistem operasi.
- Crypto ransomware: Mengenkripsi file-file tertentu atau seluruh drive.
- Screen locker ransomware: Mengunci layar dan menampilkan pesan tebusan.

13.2. Proses melaporkan insiden

Proses pelaporan insiden keamanan dapat bervariasi tergantung pada organisasi, jenis insiden, dan kebijakan yang berlaku. Setiap perusahaan idealnya memiliki prosedur yang jelas dan terstruktur untuk menangani laporan insiden. Karyawan juga berperan penting dalam mendeteksi insiden, misalnya ketika menemukan email phishing, file yang tidak dikenal, atau kinerja sistem yang tidak biasa. Namun, secara umum, berikut adalah langkah-langkah yang sering dilakukan:

Gambar 13.1 Alur Pelaporan Insiden
(Sumber: Penulis)

- Saluran Pelaporan: Ada berbagai saluran pelaporan yang bisa digunakan, seperti:
    - Hotline: Nomor telepon khusus untuk melaporkan insiden.

    - Email: Alamat email khusus untuk laporan insiden.
    - Formulir Online: Formulir yang dapat diisi secara online untuk melaporkan insiden.
    - Sistem Tiket: Sistem untuk membuat dan melacak tiket insiden.
- Informasi yang Harus Dilaporkan:
    - Waktu dan tanggal kejadian
    - Deskripsi detail tentang insiden
    - Sistem atau jaringan yang terkena dampak
    - Bukti yang mendukung (log, screenshot, dll.)
    - Dampak potensial atau aktual dari insiden

13.3. Tindakan Saat Terjadi Serangan

Isolasi perangkat yang terinfeksi adalah tindakan memutuskan atau membatasi akses perangkat yang terinfeksi dari jaringan perusahaan atau perangkat lain untuk mencegah penyebaran malware atau ancaman siber lainnya. Ketika sebuah perangkat terinfeksi, sangat penting untuk segera memutus hubungan perangkat tersebut dari jaringan agar serangan tidak menyebar ke sistem atau perangkat lainnya.

Langkah-Langkah Isolasi Perangkat yang Terinfeksi

- Putuskan koneksi dari jaringan dengan mencabut kabel atau mematikan Wi-Fi dan data seluler.
- Nonaktifkan semua perangkat eksternal, seperti USB dan Bluetooth.

- Laporkan situasi ke tim IT atau keamanan siber, menjelaskan infeksi dan langkah yang diambil.
- Hindari menggunakan perangkat lebih lanjut untuk mencegah perburukan kondisi.
- Gunakan mode aman jika diizinkan oleh tim IT untuk mencegah malware aktif.
- Perbarui antivirus dan gunakan perangkat pemulihan yang disediakan tim IT.
- Jangan sambungkan perangkat ke jaringan sebelum memastikan sudah bersih dari malware.
- Ikuti rekomendasi pencegahan dari tim IT setelah perangkat pulih, seperti memperbarui software dan mengaktifkan firewall.

13.4. Contoh Prosedur Insiden Keamanan

Insiden Ransomware di Perusahaan Kesehatan

Deskripsi: Sebuah rumah sakit besar mengalami serangan ransomware yang mengenkripsi data pasien dan sistem IT mereka setelah seorang karyawan mengklik tautan dalam email phishing. Sistem yang terinfeksi tidak dapat diakses, mengganggu layanan medis dan mengancam keselamatan pasien.

Penerapan Prosedur Insiden:

- Deteksi dan Identifikasi: Tim IT segera mendeteksi aktivitas mencurigakan di jaringan. Mereka melakukan analisis dan menemukan bahwa ransomware menginfeksi beberapa server.
- Respons Awal: Tim keamanan segera memutuskan untuk mematikan koneksi jaringan untuk mencegah penyebaran

lebih lanjut dan memberitahukan manajemen tentang insiden tersebut.

- Penilaian Dampak: Dalam waktu singkat, tim melakukan penilaian untuk menentukan sejauh mana data dan sistem terpengaruh.
- Pembersihan dan Pemulihan: Data yang tidak terinfeksi diidentifikasi, dan tim melakukan pemulihan dari cadangan untuk meminimalkan kerugian. Mereka juga melakukan analisis forensik untuk memahami bagaimana serangan terjadi.
- Pencegahan Masa Depan: Setelah pemulihan, rumah sakit memperkuat pelatihan keamanan siber untuk karyawan, menerapkan solusi keamanan tambahan seperti endpoint detection and response (EDR), dan memperbarui proses cadangan untuk memastikan bahwa data penting selalu dilindungi.

Pencurian Data Melalui Insider Threat di Perusahaan Keuangan

Deskripsi: Seorang karyawan senior di sebuah bank memutuskan untuk mencuri informasi rekening pelanggan untuk dijual ke pihak ketiga. Data ini diambil secara bertahap dan tidak terdeteksi untuk waktu yang lama.

Penerapan Prosedur Insiden:

- Deteksi: Tim keamanan akhirnya mencurigai adanya aktivitas tak biasa terkait akses data. Mereka menggunakan sistem pemantauan untuk melakukan analisis log dan menemukan pola akses yang mencurigakan.

- Investigasi: Setelah memverifikasi adanya pencurian data, tim keamanan melakukan penyelidikan menyeluruh untuk yang melibatkan audit log dan percakapan email karyawan.
- Tindakan Taktis: Karyawan tersebut segera diberhentikan, dan akses ke semua sistem perusahaan untuk karyawan tersebut dicabut. Investigasi dibawa ke jalur hukum.
- Komunikasi kepada Pelanggan: Bank segera menginformasikan kepada pelanggan bahwa data mereka telah terkompromikan dan menawarkan solusi untuk mengatasi masalah tersebut, termasuk pemantauan kredit gratis.
- Perbaikan Prosedur: Perusahaan memperbarui kebijakan keamanan dan kontrol akses dengan menerapkan segmented access controls berdasarkan prinsip least privilege. Mereka juga menerapkan program pelatihan dan kesadaran keamanan yang lebih ketat untuk semua karyawan.

Phishing dan Pencurian Informasi Login di Perusahaan Logistik

Deskripsi: Karyawan di perusahaan logistik menerima email phishing yang tampak resmi, mengarahkan mereka untuk memasukkan informasi login. Karyawan yang tidak curiga beberapa di antaranya terjebak, dan data login mereka dicuri.

Penerapan Prosedur Insiden:

- Deteksi: Beberapa karyawan melaporkan bahwa mereka tidak bisa mengakses sistem setelah memberikan informasi login. Tim IT segera menyelidiki dan menemukan bahwa beberapa akun telah disusupi.

- Tindakan Kontrol: Semua akun yang terinfeksi segera dinonaktifkan, dan karyawan yang terlibat diberitahu. Tim IT juga melakukan pemindaian untuk memastikan tidak ada malware yang terpasang di jaringan.
- Pembersihan: Selain menonaktifkan akun yang terkompromi, tim memastikan bahwa semua kata sandi diperbarui dan kebijakan keamanan yang lebih ketat diimplementasikan.
- Pendidikan Pengguna: Perusahaan kemudian meluncurkan program pelatihan khusus untuk semua karyawan tentang cara mengenali dan merespons ancaman phishing.
- Analisis Keamanan: Di akhir proses, perusahaan melakukan analisis keseluruhan sistem untuk menilai apakah ada celah lain yang mungkin diminati oleh penyerang.

Kasus-kasus di atas menggambarkan berbagai cara perusahaan menerapkan prosedur insiden keamanan untuk merespons dan menangani berbagai jenis insiden. Reaksi cepat, komunikasi yang efektif, serta evaluasi dan perbaikan berkelanjutan adalah kunci untuk mengurangi dampak insiden keamanan dan mencegah kejadian serupa di masa depan. Setiap langkah yang diambil dalam siklus respons insiden membantu dalam membangun budaya keamanan yang lebih kuat di perusahaan.

CHAPTER 14
PRAKTIK KEAMANAN DI TEMPAT KERJA

Materi:

- Penggunaan perangkat USB dengan aman
- Mengunci komputer saat tidak digunakan
- Menghindari kebocoran data fisik (misalnya, melalui dokumen cetak)

Durasi:

- 1 Jam

Tujuan:

- Mendorong kebiasaan aman di tempat kerja untuk mengurangi risiko kebocoran data.

Target Pembaca:

- Masyarakat, Mahasiswa & Karyawan Non-IT

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

14.1. Pendahuluan Praktik Keamanan

Keamanan di tempat kerja adalah tanggung jawab semua karyawan, bukan hanya tim IT. Praktik-praktik sederhana seperti menggunakan password yang kuat, tidak meninggalkan komputer tanpa dikunci, dan berhati-hati saat menggunakan perangkat USB dapat membantu melindungi data dan aset perusahaan dari ancaman siber. Praktik keamanan di tempat kerja merupakan serangkaian upaya untuk menciptakan lingkungan kerja yang aman, sehat, dan bebas dari bahaya. Tujuan utama dari praktik ini adalah untuk mencegah terjadinya kecelakaan kerja, cedera, penyakit akibat kerja, dan kerugian material.

Manfaat kita menerapkan praktik keamanan di tempat kerja atau perusahaan adalah sebagai berikut:

- Perlindungan Data Sensitif
- Mencegah Kerugian Finansial
- Mempertahankan Reputasi Perusahaan
- Kepatuhan terhadap Regulasi

14.1.1. Penggunaan Aman Perangkat USB

Perangkat USB sering kali menjadi sumber infeksi malware karena sifatnya yang portabel dan sering berpindah-pindah antar perangkat. Virus atau malware dapat disebarkan melalui USB tanpa disadari.

Praktik Keamanan:

- Gunakan hanya perangkat USB yang terpercaya. Hindari menggunakan USB dari sumber yang tidak jelas, terutama yang diperoleh secara gratis di konferensi atau pameran.
- Scan perangkat USB sebelum digunakan. Pastikan USB di-scan dengan software antivirus sebelum membuka atau menyalin file.

- Jangan pasang USB yang tidak dikenal. Jika Anda menemukan USB yang tidak Anda ketahui asal-usulnya, serahkan kepada tim IT daripada memasangnya langsung ke komputer.

Gunakan perangkat USB yang dilengkapi dengan enkripsi untuk melindungi data yang sensitif.

14.1.2. Mengunci Komputer

Mengunci komputer adalah langkah dasar untuk menjaga agar tidak ada orang lain yang bisa mengakses data atau aplikasi penting saat Anda meninggalkan meja. Praktik Keamanan sbb:

- Gunakan kombinasi tombol cepat untuk mengunci komputer. Di Windows, tekan Windows + L, dan di macOS tekan Control + Command + Q sebelum meninggalkan meja, bahkan untuk waktu singkat.
- Otomatisasi penguncian layar. Pastikan komputer dikonfigurasi untuk otomatis terkunci setelah periode tidak aktif, misalnya, 5 atau 10 menit tanpa aktivitas.
- Jangan tinggalkan perangkat tanpa pengawasan. Jika Anda harus meninggalkan ruangan, selalu pastikan perangkat terkunci atau dibawa bersama Anda.

Contoh serangan terhadap komputer atau perangkat yang tidak di kunci adalah sebagai berikut:

- Shoulder Surfing: Shoulder Surfing adalah teknik pengintaian yang dilakukan dengan cara mengamati atau

mengintip informasi sensitif yang sedang dilihat atau digunakan seseorang, biasanya di tempat umum. Praktik ini sering terjadi saat seseorang menggunakan gadget seperti smartphone atau laptop, di mana orang lain bisa melihat layar atau mengetikkan informasi pribadi, seperti kata sandi atau nomor kartu kredit.

- Quid Pro Quo: Quid Pro Quo dalam konteks keamanan siber merujuk pada jenis serangan atau teknik manipulasi di mana seorang penyerang menawarkan sesuatu yang bernilai kepada korban sebagai imbalan untuk mendapatkan informasi sensitif atau akses ke sistem. Misalnya, seorang penyerang mungkin berpura-pura menjadi teknisi IT dan menawarkan bantuan gratis untuk memperbaiki masalah pada komputer korban, namun dalam prosesnya, mereka meminta informasi login atau akses ke sistem.
- Baiting: Baiting adalah teknik dalam keamanan siber yang digunakan oleh penyerang untuk menarik perhatian korban dengan iming-iming yang menarik atau menggoda. Dalam konteks ini, penyerang biasanya menawarkan sesuatu yang menarik, seperti file gratis, perangkat lunak, atau konten eksklusif, dengan tujuan untuk mendapatkan informasi pribadi atau menginfeksi perangkat korban dengan malware.
- Tailgating: Tailgating dalam konteks keamanan siber merujuk pada situasi di mana seseorang yang tidak memiliki izin mencoba untuk memasuki area yang dilindungi dengan mengikuti orang yang berwenang. Ini sering terjadi di pintu masuk gedung, di mana seorang karyawan dapat membuka pintu untuk diri mereka sendiri, dan orang lain yang tidak dikenal mencoba untuk "mengikuti" mereka tanpa izin.

14.1.3. Menghindari Kebocoran Data Fisik

Banyak karyawan berpikir bahwa ancaman keamanan hanya datang dari dunia digital, tetapi kebocoran informasi juga bisa terjadi secara fisik, seperti membuang dokumen penting tanpa merusaknya atau meninggalkan cetakan dokumen sensitif di printer. Praktik Keamanan sbb:

- Gunakan shredders (penghancur kertas) untuk membuang dokumen penting. Jangan pernah membuang dokumen sensitif ke tempat sampah tanpa menghancurkannya terlebih dahulu.
- Periksa printer sebelum meninggalkan area. Jika Anda mencetak dokumen penting, segera ambil dari printer dan pastikan tidak meninggalkan cetakan sembarangan.
- Simpan dokumen fisik di tempat yang aman. Gunakan laci atau lemari yang terkunci untuk menyimpan dokumen sensitif di kantor.

Beberapa contoh serangan yang terkait dengan kebocoran data fisik adalah sebagai berikut:

- Pretexting dengan Kedok Servis: Pretexting dengan kedok servis adalah salah satu teknik penipuan dalam konteks keamanan siber di mana penyerang berpura-pura menjadi pihak yang sah, seperti penyedia layanan atau dukungan teknis, untuk mendapatkan informasi sensitif dari korban. Metode ini sering kali melibatkan penggunaan skenario yang meyakinkan untuk melegitimasi permintaan informasi pribadi atau rahasia.

  Dalam praktiknya, penyerang mungkin menghubungi korban melalui telepon, email, atau pesan instan, mengklaim bahwa mereka perlu memverifikasi identitas korban atau memperbaiki masalah teknis yang dialami.

Dengan cara ini, penyerang berusaha menciptakan kepercayaan dan mengalihkan perhatian korban dari potensi bahaya.

- Dumpster Diving: Dumpster diving, dalam konteks keamanan siber, merujuk pada praktik mencari informasi sensitif atau berharga dengan mengacak-acak tempat sampah atau limbah perusahaan. Pelaku kejahatan siber sering kali mencari dokumen fisik, seperti catatan bisnis, dokumen keuangan, atau data pribadi yang bisa digunakan untuk melakukan penipuan, pencurian identitas, atau serangan lainnya.

  Informasi yang ditemukan melalui dumpster diving dapat mencakup rincian login, nomor kartu kredit, atau informasi pribadi lainnya yang seharusnya dibuang dengan aman. Oleh karena itu, penting bagi perusahaan dan individu untuk menerapkan kebijakan pengelolaan limbah yang baik, termasuk penggunaan penghancur kertas dan pemusnahan data yang tepat, guna melindungi informasi sensitif dari jatuh ke tangan yang salah.

14.1.4. Keamanan Perangkat Mobile di Tempat Kerja

Perangkat mobile seperti smartphone atau tablet juga dapat menjadi sumber kebocoran data, terutama jika terhubung ke jaringan kantor atau digunakan untuk mengakses email kerja. Kehilangan atau pencurian perangkat bisa mengakibatkan akses tidak sah ke sistem perusahaan.

Praktik Keamanan:

- Selalu gunakan kunci layar atau kata sandi. Pastikan perangkat mobile yang digunakan untuk bekerja dilindungi dengan PIN, password, atau pengenalan biometrik.

- Enkripsi perangkat. Pastikan perangkat mobile dienkripsi sehingga jika perangkat hilang atau dicuri, data di dalamnya tetap terlindungi.
- Hindari penggunaan perangkat mobile pada Wi-Fi publik tanpa VPN. Gunakan VPN saat mengakses sistem perusahaan atau data penting melalui jaringan publik.

14.1.5. Menghindari Ancaman Email Berbahaya

Email adalah salah satu metode paling umum yang digunakan penjahat siber untuk menyebarkan malware, phising, atau melakukan penipuan. Karyawan non-IT sering kali menjadi target karena kurangnya kesadaran terhadap ancaman email.

Praktik Keamanan:

- Waspadai email yang mencurigakan. Jangan klik tautan atau unduh lampiran dari pengirim yang tidak dikenal atau yang terlihat mencurigakan.
- Verifikasi sumber email sebelum mengambil tindakan. Jika email tampaknya berasal dari atasan atau rekan kerja, tetapi kontennya mencurigakan, verifikasi melalui saluran lain, seperti telepon.
- Jangan bagikan informasi sensitif melalui email. Hindari mengirimkan data sensitif seperti password atau nomor kartu kredit melalui email yang tidak terenkripsi.

14.2. Simulasi

Challenge

Ada 4 simulasi yang harus karyawan lakukan. Berikut skenarionya :

1. **Simulasi Penggunaan Aman Perangkat USB**
   Peserta diberikan USB dan diminta untuk menentukan langkah-langkah yang harus dilakukan sebelum menggunakan USB tersebut di komputer kerja.

2. **Latihan Mengunci Komputer**
   Lakukan penguncian otomatis di perangkat masing-masing dan ajarkan cara mengaktifkan fitur ini.

3. **Tugas Pengamanan Perangkat Mobile**
   Tinjau pengaturan keamanan perangkat mobile masing-masing dan terapkan fitur-fitur keamanan yang direkomendasikan.

4. **Studi Kasus Printer**
   Diskusi kelompok tentang risiko yang dapat terjadi jika dokumen sensitif tertinggal di printer bersama.

CHAPTER 15
STUDI KASUS DAN DISKUSI

Materi:

- Simulasi serangan phishing
- Kasus pelanggaran keamanan dan bagaimana mengatasinya
- Diskusi dan tanya jawab

Durasi:

- 1 Jam

Tujuan:

- Menyelesaikan beberapa studi kasus dan diskusi untuk memperkuat pemahaman karyawan tentang risiko dan tindakan mitigasi.

Target Pembaca:

- Masyarakat, Mahasiswa & Karyawan Non-IT

Penulis/Trainer:

- Doddy Ferdiansyah, R. Rizal Isnanto, Jatmiko E. Suseno

15.1. Studi Kasus: Skenario 1

Seorang karyawan bernama Budi bekerja sebagai staf administrasi di perusahaan. Suatu hari, Budi menerima email yang terlihat seperti berasal dari direktur keuangan, yang meminta informasi login akun sistem keuangan perusahaan. Email tersebut menyertakan tautan yang meminta Budi untuk memasukkan kredensialnya ke halaman login yang terlihat sah. Karena Budi tidak ingin mengecewakan direktur, dia segera memberikan informasi tersebut tanpa verifikasi.

Apa kesalahan yang dilakukan Budi dalam skenario ini?

Apa yang bisa Budi lakukan untuk memastikan keamanan sebelum menindaklanjuti permintaan tersebut?

Bagaimana karyawan non-IT dapat mengenali email phishing dan melaporkannya ke tim IT?

15.2. Studi Kasus: Skenario 2

Seorang karyawan bernama Dita bekerja sebagai staf di departemen pemasaran di perusahaan. Suatu hari, Dita menerima email yang tampaknya berasal dari manajernya, meminta konfirmasi data pemasaran yang sensitif dan menginstruksikan Dita untuk mengklik link yang ada di email tersebut. Meskipun terlihat resmi, email tersebut sebenarnya merupakan phishing. Dan karena merasa urgent email tersebut, maka Dita mengklik link tersebut dan dalam beberapa saat, data-data di perusahaan hilang.

Apa kesalahan yang dilakukan Dita dalam skenario ini?

Apa yang bisa Dita lakukan untuk memastikan keamanan sebelum menindaklanjuti permintaan tersebut?

15.3. Studi Kasus: Skenario 3

Siti, seorang pegawai di departemen akuntansi, menggunakan kata sandi yaitu Sit1@1995 untuk mengakses sistem keuangan perusahaan. Suatu ketika, seorang rekan kerja yang penasaran mencoba masuk ke akun Siti hanya dengan menggunakan kata sandi tersebut dan berhasil mengakses sistem.

Insiden ini segera memicu alarm di seluruh perusahaan. Departemen IT segera diberitahu dan langkah-langkah keamanan darurat diaktifkan untuk mencegah akses yang tidak sah lebih lanjut.

Apa kesalahan yang dilakukan Siti dalam skenario ini?

Apa yang bisa Siti lakukan agar akunnya bisa lebih aman?

15.4. Studi Kasus: Skenario 4

Dani, seorang pegawai di bidang logistik, meninggalkan laptop kerjanya di dalam mobil saat pergi untuk makan siang. Sayangnya, mobil tersebut dicuri, sehingga data sensitif perusahaan yang ada di laptop itu berisiko untuk bocor ke publik. Kejadian ini membuat Dani merasa sangat cemas dan bertanggung jawab atas insiden tersebut.

Apa yang bisa dilakukan Dani untuk memastikan keamanan perangkatnya di masa depan?

Apa langkah-langkah yang harus diambil setelah insiden kehilangan perangkat untuk meminimalkan dampak pada data perusahaan?

15.5. Studi Kasus: Skenario 5

Rina, yang bekerja di departemen komunikasi, membagikan informasi mengenai peluncuran produk baru perusahaan di akun media sosial pribadinya. Meski Rina merasa bangga dengan pencapaian perusahaan, ia tidak menyadari bahwa informasi tersebut adalah rahasia.

Apa risiko yang terkait dengan pengungkapan informasi rahasia di media sosial?

Bagaimana Rina bisa menghindari kejadian serupa di masa mendatang?

Apa saja kebijakan yang perlu diterapkan oleh perusahaan terkait penggunaan media sosial oleh karyawan?

DAFTAR PUSTAKA

Al-Matouq, H., Mahmood, S., Alshayeb, M., & Niazi, M. (2020). A Maturity Model for Secure Software Design: A Multivocal Study. In IEEE Access (Vol. 8, p. 215758). Institute of Electrical and Electronics Engineers.

Alnafjan, K., Hussain, T., Ullah, H., & Paracha, Z. ul haq. (2013). Comparative Analysis and Evaluation of Software Vulnerabilities Testing Techniques. In World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering (Vol. 7, Issue 6, p. 687).

Anjum, M., Kapur, P. K., Agarwal, V., & Khatri, S. K. (2020). Evaluation and Selection of Software Vulnerabilities. In International Journal of Reliability Quality and Safety Engineering (Vol. 27, Issue 5, p. 2040014). World Scientific.

Anwer, F., Nazir, M., & Mustafa, K. (2016). Security Testing. In F. Anwer, M. Nazir, & K. Mustafa, Springer eBooks (p. 35). Springer Nature.

Author Karen Scarfone; Author Murugiah Souppaya; Author Amanda Cody; Author Angela Orebaugh. (2023). NIST Special Publication (SP) 800-115, Technical Guide to Information Security Testing and Assessment.

Author Richard Kissel; Author Kevin Stine; Author Matthew Scholl; Author Hart Rossman; Author Jim Fahlsing; Author Jessica Gulick. (2023). NIST Special Publication (SP) 800-64 Rev. 2 (Withdrawn), Security Considerations in the System Development Life Cycle.

Bacudio, A. G., Yuan, X., Chu, B. T. B., & Jones, M. M. (2011). An Overview of Penetration Testing. In International

Journal of Network Security & Its Applications (Vol. 3, Issue 6, p. 19).

Banerjee, C., & Pandey, S. K. (2009). Software Security Rules, SDLC Perspective. In arXiv (Cornell University). Cornell University.

Bertrand, J. T., Brown, J., & Ward, V. (1992). Techniques for Analyzing Focus Group Data. In Evaluation Review (Vol. 16, Issue 2, p. 198). SAGE Publishing.

Black, P. E., Guttman, B., & Okun, V. (2021). Guidelines on minimum standards for developer verification of software.

Broad, J. (2013). System Development Life Cycle (SDLC). In Elsevier eBooks (p. 39). Elsevier BV.

Curry, L., Nembhard, I. M., & Bradley, E. H. (2009). Qualitative and Mixed Methods Provide Unique Contributions to Outcomes Research [Review of Qualitative and Mixed Methods Provide Unique Contributions to Outcomes Research]. Circulation, 119(10), 1442. Lippincott Williams & Wilkins.

Cyber Security White Papers. (2023). https://www.sans.org/white-papers/

Davis, N., Humphrey, W. S., Redwine, S. T., Zibulski, G., & McGraw, G. (2004). Processes for producing secure software. In IEEE Security & Privacy (Vol. 2, Issue 3, p. 18). Institute of Electrical and Electronics Engineers.

Dodson, D., Souppaya, M., & Scarfone, K. (2020). Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF).

Duggleby, W. (2005). What About Focus Group Interaction Data? In W. Duggleby, Qualitative Health Research (Vol. 15, Issue 6, p. 832). SAGE Publishing.

Eian, I. C., Yong, L. K., Li, M. Y. X., Hasmaddi, N. A. B. N., & Fatima-tuz-Zahra. (2020). Integration of Security Modules

in Software Development Lifecycle Phases. In arXiv (Cornell University). Cornell University.

Fariduddin, F., Nasution, A., & Weistroffer, H. R. (2009). Documentation in Systems Development: A Significant Criterion for Project Success.

Ferdiansyah, D., Isnanto, R. R., & Suseno, J. E. (2023). Organizational indicators on startup software for implementing secure software development lifecycle (SSDL): A systematic literature review. In AIP conference proceedings. American Institute of Physics.

Ferdiansyah, D., Isnanto, R. R., & Suseno, J. E. (2023). Strategy Indicators for Secure Software Development Lifecycle in Software Startups Based on Information Security Governance. In Journal of Internet Services and Information Security (Vol. 13, Issue 4, p. 104).

Futcher, L., & Solms, R. von. (2007). SecSDM: A Model for Integrating Security into the Software Development Life Cycle. In IFIP advances in information and communication technology (p. 41). Springer Science+Business Media.

Futcher, L., & Solms, R. von. (2008). Guidelines for secure software development (By L. Futcher & R. von Solms; Vol. 48, p. 56).

Fuzzing. (2023). https://www.microsoft.com/en-us/research/uploads/prod/2020/03/Fuzzing-101-CACM2020.pdf

Hatfield, J. M. (2019). Virtuous human hacking: The ethics of social engineering in penetration-testing. In Computers & Security (Vol. 83, p. 354). Elsevier BV.

Hu, V. C., Grance, T., Ferraiolo, D., & Kühn, D. (2014). An Access Control Scheme for Big Data Processing.

Jeong, S., Kang, S.-Y., & Kim, S. (2020). Towards Security-by-design in Automotive Development Process.

Joint Task Force. (2018). NIST Special Publication (SP) 800-37 Rev. 2, Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy.

Kakareka, A. (2009). What Is Vulnerability Assessment? In Elsevier eBooks (p. 383). Elsevier BV.

Kates, J. (2001). Security Testing Is Not All the Same: A Reference Taxonomy. In Auerbach Publications eBooks (p. 607).

Kaur, D., Kaur, P., Patel, R. B., & Singh, B. P. (2011). Software Development Life Cycle Security Issues. In AIP conference proceedings. American Institute of Physics.

Kawulich, B. (2005). Participant Observation as a Data Collection Method. In Forum qualitative Sozialforschung (Vol. 6, Issue 2). Freie Universität Berlin.

Khan, S., Parkinson, S., & Crampton, A. (2017). A Multi-layered Cloud Protection Framework (By S. Khan, S. Parkinson, & A. Crampton; Vol. 7, p. 233).

Kissel, R., Stine, K., Scholl, M., Rossman, H., Fahlsing, J., & Gulick, J. (2008). Security considerations in the system development life cycle.

Kudriavtseva, A., & Gadyatskaya, O. (2022). Secure Software Development Methodologies: A Multivocal Literature Review. In arXiv (Cornell University). Cornell University.

Kumar, S., Mahajan, R., Kumar, N., & Khatri, S. K. (2017). A study on web application security and detecting security vulnerabilities (p. 451).

Lingham, A. D., Kin, N. T. K., Chen, J., Loong, C. H., & Fatima-tuz-Zahra. (2020). Implementation of Security Features in Software Development Phases. In arXiv (Cornell University). Cornell University.

Manes, V. J. M., Han, H.-S., Han, C., Cha, S. K., Egele, M., Schwartz, E. J., & Woo, M. (2018). The Art, Science, and

Engineering of Fuzzing: A Survey. In arXiv (Cornell University). Cornell University.

Manjhi, A., Ailamaki, A., Maggs, B. M., Mowry, T. C., Olston, C., & Tomasic, A. (2006). Simultaneous scalability and security for data-intensive web applications.

Marconato, G., Kaâniche, M., & Nicomette, V. (2012). A Vulnerability Life Cycle-Based Security Modeling and Evaluation Approach. In The Computer Journal (Vol. 56, Issue 4, p. 422). Oxford University Press.

Maskur, A. F., & Asnar, Y. (2019). Static Code Analysis Tools with the Taint Analysis Method for Detecting Web Application Vulnerability.

Mkpong-Ruffin, I., Umphress, D., Hamilton, J. A., & Gilbert, J. E. (2007). Quantitative software security risk assessment model (p. 31).

Mohino, de V., Higuera, B., Higuera, B., & Montalvo, J. A. S. (2019). The Application of a New Secure Software Development Life Cycle (S-SDLC) with Agile Methodologies. In Electronics (Vol. 8, Issue 11, p. 1218). Multidisciplinary Digital Publishing Institute.

Mudiyanselage, A. K., & Pan, L. (2017). Security test MOODLE: a penetration testing case study. In International Journal of Computers and Applications (Vol. 42, Issue 4, p. 372). Taylor & Francis.

Murugiah Souppaya (NIST). (2022). NIST Special Publication (SP) 800-218, Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities.

Nagpure, S., & Kurkure, S. (2017). Vulnerability Assessment and Penetration Testing of Web Application (p. 1).

Naik, N. A., Kurundkar, G. D., Khamitkar, S., & Kalyankar, N. V. (2009). Penetration Testing: A Roadmap to Network Security. In arXiv (Cornell University). Cornell University.

Nelson, A., Dinolt, G. W., Michael, J., & Shing, M. (2011). A security and usability perspective of cloud file systems (By A. Nelson, G. W. Dinolt, J. Michael, & M. Shing; Vol. 8, p. 161).

Nguyen-Duc, A., Do, M.-V., Luong-Hong, Q., Nguyen-Khac, K., & Truong-Anh, H. (2021). On the combination of static analysis for software security assessment. https://arxiv.org/pdf/2103.08010v2.pdf

OWASP Top 10:2021. (2021). https://owasp.org/Top10/

Phanindra, A. R., Narasimha, V. B., & PhaniKrishna, Ch. V. (2018). A Review on Application Security Management Using Web Application Security Standards. Advances in Intelligent Systems and Computing, 477. Springer Nature.

Qu, Z., Alam, S., Chen, Y., Zhou, X., Hong, W., & Riley, R. (2017). DyDroid: Measuring Dynamic Code Loading and Its Security Implications in Android Applications.

Ramírez, A., Aiello, A., & Lincke, S. (2020). A Survey and Comparison of Secure Software Development Standards.

Reichert, B. M., & Obelheiro, R. R. (2022). An Integrity-Focused Threat Model for Software Development Pipelines. In arXiv (Cornell University). Cornell University.

Risk management framework for information systems and organizations: (2018). https://doi.org/10.6028/nist.sp.800-37r2

Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008). Technical guide to information security testing and assessment.

Scott, D., & Sharp, R. R. (2002). Developing secure Web applications. In IEEE Internet Computing (Vol. 6, Issue 6, p. 38). IEEE Computer Society.

Secure By Design. (2023). https://www.cisa.gov/sites/default/files/2023-10/SecureByDesign_1025_508c.pdf

Security Training Class Video Sample SDL Documents. (2023). https://download.microsoft.com/download/8/1/6/816C597A-5592-4867-A0A6-A0181703CD59/Microsoft_Press_eBook_TheSecurityDevelopmentLifecycle_PDF.pdf

Shebli, H. M. Z. A., & Beheshti, B. D. (2018). A study on penetration testing process and tools.

Souppaya, M., Scarfone, K., & Dodson, D. (2022). Secure Software Development Framework (SSDF) version 1.1 : https://doi.org/10.6028/nist.sp.800-218

The Top Cyber Security Risks. (2022). https://pdfslide.us/documents/sans-the-top-cyber-security-risks-the-top-cyber-security-riskspdf-the-top-cyber.html

Tian-yang, G., Yin-sheng, S., & You-yuan, F. (2010). Research on Software Security Testing. In World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering (Vol. 4, Issue 9, p. 1446).

Tompkins, F. G., & Rice, R. S. (1986). Integrating security activities into the software development life cycle and the software quality assurance process. In Computers & Security (Vol. 5, Issue 3, p. 218). Elsevier BV.

Weixiong, Y., Dozono, K., Lee, R., Seng, A. K. S., & Zahra, F. (2020). Securing Software Systems.

Whitehead, D., & Disler, R. (2020). Common qualitative methods. http://ecite.utas.edu.au/140285

Winkler, V. (2011). Securing the Cloud: Architecture. https://www.sciencedirect.com/science/article/pii/B9781597492900004X

Ye, N. (2008). Assets, Vulnerabilities and Threats of Computer and Network Systems (p. 3).

Zhen, L., Xiong, Z., & Tu, K. Q. (2014). Assessment Model and Method Research of Information Security Risk. In Applied

Mechanics and Materials (Vol. 496, p. 2170). Trans Tech Publications.

BIODATA PENULIS

Doddy Ferdiansyah, ST., MT, saat ini sebagai dosen tetap di Program Studi Teknik Informatika, Fakultas Teknik, Universitas Pasundan, Bandung, Indonesia. Perjalanan akademik di perguruan tinggi dimulai setelah lulus program S1 Teknik Informatika di Universitas Pasundan, kemudian melanjutkan sekolah di program studi Magister Teknik Informatika di Universitas Langlangbuana di Bandung dengan mengambil jurusan Cyber Security. Dan saat ini sedang menyelesaikan program Doktoral Sistem Informasi di Universitas Diponegoro, Semarang dalam bidang Information Security Governance terhadap Secure Software Development Lifecycle. Untuk riset utama yang dilakukan dia adalah di bidang cyber security, digital forensic, dan incident response. Hal ini sudah dilakukan sejak S1 karena minat dan passion yang sangat tinggi terhadap bidang keamanan siber. Selain itu, dia juga aktif di media sosial sebagai content creator dan edukator di bidang cyber security baik di Youtube, Instagram, dan TikTok. Hal ini dilakukan untuk terus memberikan edukasi dan awareness kepada masyarakat terkait bahaya dari ancaman-ancaman di dunia siber saat ini dan di masa yang akan datang secara konsisten.

Sinopsis

Dalam era digital yang terus berkembang, keamanan aplikasi menjadi salah satu aspek paling krusial dalam pengembangan perangkat lunak. Buku "Membangun Aplikasi yang Aman: Secure Software Development Lifecycle (SSDL)" menawarkan panduan komprehensif untuk mengintegrasikan praktik keamanan di setiap fase pengembangan aplikasi. Ditulis oleh Doddy Ferdiansyah, R. Rizal Isnanto, dan Jatmiko E. Suseno, buku ini dirancang untuk membantu pengembang, pemangku kepentingan, dan institusi pendidikan dalam memahami pentingnya dan implementasi SSDL.

Buku ini dimulai dengan dasar-dasar pengenalan Secure SDLC, membedah setiap tahapan dari pengumpulan kebutuhan hingga deployment dan pemeliharaan, dengan fokus pada implementasi keamanan sejak dini. Dengan pendekatan sistematis, pembaca akan mempelajari cara merancang, mengimplementasikan, dan menguji aplikasi agar tidak hanya fungsional tetapi juga aman dari ancaman dan kerentanan potensial.

Setiap bab dilengkapi dengan studi kasus, diskusi teoritis, dan praktis yang menunjukkan skenario nyata dan cara menghadapi berbagai tantangan keamanan dalam pengembangan perangkat lunak. Pembahasan meliputi topik seperti autentikasi pengguna, kontrol akses, manajemen risiko, hingga kebijakan keamanan dan strategi penerapan di lingkungan startup. "Membangun Aplikasi yang Aman" tidak hanya memberikan teori tetapi juga praktek dan tools yang dibutuhkan untuk memperkuat keamanan perangkat lunak, menjadikannya sumber daya yang sangat berharga bagi siapa saja yang terlibat dalam siklus hidup pengembangan perangkat lunak.

Diterbitkan oleh

ASOSIASI DOKTOR Sistem Informasi INDONESIA

ISBN 978-623-10-7157-6

9 786231 071576