



2025  
2026

# Laporan Analisis dan Implementasi Aplikasi Audit Keamanan Sistem serta Deteksi Anomali Berbasis Pembelajaran Mesin

Tugas Akhir Semester – Peretasan Etis A

*Wilfridus Bambang Triadi Handaya, S.T., M.Cs.*



## Anggota Kelompok

1. Arwindo Sedy Pratama (230712555)
2. Yonatan Adi Cahyoningrat (230712440)
3. Gracia Putri Aura (230712515)
4. Gretelia Faustine (230712322)
5. Velin Ceria Resminawati (230712478)

Program Studi Informatika  
Fakultas Teknologi Industri  
Universitas Atma Jaya Yogyakarta

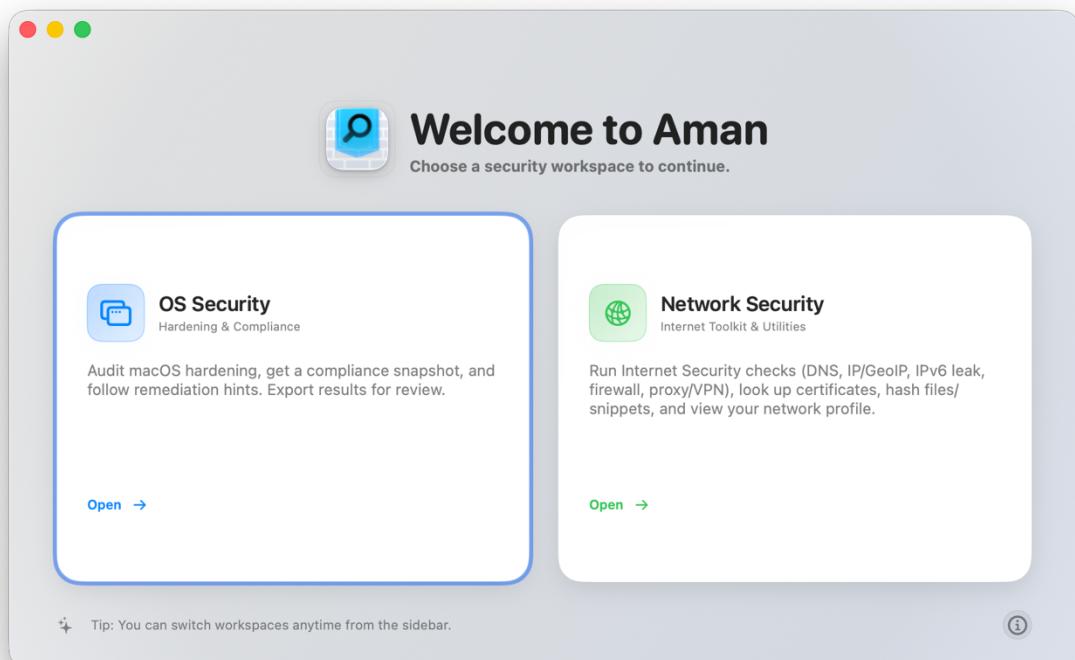
<b>BAB I Pendahuluan .....</b>	<b>3</b>
A. Latar Belakang .....	3
B. Rumusan Masalah.....	4
C. Tujuan Pembuatan Aplikasi.....	4
D. Manfaat Aplikasi.....	4
E. Batasan Masalah.....	5
F. Ruang Lingkup Laporan.....	5
G. Metode Pengembangan .....	6
<b>BAB II Landasan Teori .....</b>	<b>7</b>
A. Pengertian Aplikasi.....	7
B. Konsep Dasar Cybersecurity Assessment.....	7
C. Konsep Machine Learning dalam Keamanan Sistem .....	8
D. Arsitektur Aplikasi.....	9
E. Etika Penggunaan dalam Peretasan Etis .....	10
<b>BAB III Perancangan Sistem .....</b>	<b>12</b>
A. Gambaran Umum Sistem.....	12
B. Arsitektur Utama .....	14
C. Diagram Alur Proses Deteksi Anomali.....	16
D. Desain Antarmuka Pengguna .....	19
E. Deskripsi Fitur-Fitur Utama.....	22
<b>BAB IV Implementasi Aplikasi dan Hasil .....</b>	<b>24</b>
A. Lingkungan Pengembangan .....	24
B. Implementasi Fitur-Fitur Aplikasi.....	24
C. Integrasi Antarmuka Pengguna .....	26
D. Alur Penggunaan Aplikasi oleh Pengguna .....	27
E. Keunggulan dan Kelemahan Aplikasi .....	28
<b>BAB V Penutup .....</b>	<b>29</b>
A. Kesimpulan.....	29
B. Saran Pengembangan .....	29
<b>Daftar Pustaka .....</b>	<b>31</b>

# BAB I Pendahuluan

## A. Latar Belakang

Latar belakang pengembangan aplikasi *Aman* didasari oleh kebutuhan kritis untuk memperkuat keamanan ekosistem macOS. Meskipun dikenal memiliki fitur keamanan bawaan yang handal, risiko keamanan tetap muncul dari konfigurasi yang kurang optimal dan ancaman jaringan yang semakin canggih. Banyak alat audit keamanan yang tersedia saat ini bersifat terpisah, sulit digunakan, atau bergantung pada layanan *cloud* yang berpotensi mengurangi privasi pengguna. **Aplikasi *Aman* hadir sebagai solusi terintegrasi dan privat** untuk menjembatani kesenjangan antara keamanan bawaan sistem dan praktik penguatan keamanan yang komprehensif, dengan poin-poin sebagai berikut:

- Perangkat macOS semakin banyak digunakan untuk pekerjaan kritis, pengelolaan data sensitif, dan aktivitas daring sehari-hari.
- Konfigurasi yang tidak tepat, seperti berbagi *file*, port terbuka, atau kebijakan kata sandi lemah, tetap dapat membuka celah bagi ancaman.
- Aplikasi *Aman* dirancang untuk menjawab kebutuhan akan aplikasi audit keamanan sistem dan deteksi anomali jaringan yang terintegrasi, mudah digunakan, dan beroperasi secara lokal di perangkat pengguna.



Gambar 1 Tampilan utama aplikasi / landing page

## B. Rumusan Masalah

Untuk memastikan pengembangan aplikasi *Aman* berjalan terfokus dan mencapai tujuan yang diinginkan, perlu dirumuskan serangkaian pertanyaan kunci yang akan dijawab melalui implementasi fitur-fitur utama. Rumusan masalah ini mencakup aspek audit sistem, pengungkapan paparan jaringan, hingga penerapan teknologi cerdas dalam analisis data, antara lain:

- Bagaimana merancang sebuah aplikasi yang mampu menilai *posture* keamanan sistem operasi macOS secara menyeluruh dan terstruktur?
- Bagaimana mengidentifikasi paparan jaringan (seperti port dan layanan yang terbuka pada jaringan lokal) tanpa mengorbankan privasi pengguna?
- Bagaimana menerapkan konsep *machine learning* dan analitik statistik untuk memberikan deteksi anomali lalu lintas jaringan yang dapat dipahami oleh pengguna non-ahli?
- Bagaimana menyajikan informasi teknis tersebut dalam bentuk antarmuka yang ringkas, interaktif, dan mudah ditindaklanjuti?

## C. Tujuan Pembuatan Aplikasi

Tujuan pembuatan aplikasi *Aman* merupakan cerminan langsung dari rumusan masalah yang diajukan, sekaligus menjadi tolok ukur keberhasilan implementasi. Secara umum, aplikasi ini bertujuan untuk menghasilkan sebuah platform yang holistik dalam audit keamanan dan deteksi anomali. Tujuan-tujuan spesifik tersebut adalah:

- Menghasilkan aplikasi audit keamanan sistem yang dapat memeriksa berbagai aspek konfigurasi macOS, mulai dari enkripsi disk, integritas sistem, kebijakan akun, hingga layanan jaringan.
- Menyediakan modul pemetaan jaringan dan pemindaian port untuk mengungkap perangkat serta layanan yang aktif pada jaringan lokal pengguna.
- Mengimplementasikan *pipeline* deteksi anomali berbasis *machine learning* dan statistik sehingga pola lalu lintas yang tidak lazim dapat diidentifikasi lebih dini.
- Menyediakan laporan dan ringkasan hasil yang mudah dipahami sehingga pengguna dapat segera mengambil langkah perbaikan.

## D. Manfaat Aplikasi

Kehadiran aplikasi *Aman* diharapkan dapat memberikan kontribusi signifikan di berbagai sektor, baik bagi pengguna individu, profesional keamanan, maupun ranah akademis. Manfaat yang diharapkan terbagi berdasarkan target audiens, di antaranya:

- **Bagi pengguna individu**

mengetahui kondisi keamanan perangkatnya dan mendapatkan panduan singkat mengenai tindakan perbaikan yang disarankan.

- **Bagi pengelola TI dan keamanan**

memperoleh gambaran konsolidasi mengenai konfigurasi keamanan dan paparan jaringan macOS yang dikelola, serta bahan laporan audit internal.

- **Bagi dunia pendidikan dan penelitian**

menjadi studi kasus implementasi nyata integrasi audit sistem, pemetaan jaringan, dan deteksi anomali berbasis *machine learning* pada lingkungan macOS.

## E. Batasan Masalah

Guna menjaga fokus proyek, memastikan implementasi yang mendalam, dan mengelola kompleksitas pengembangan, aplikasi *Aman* menerapkan serangkaian batasan fungsional dan teknis. Batasan-batasan ini dirancang agar proyek dapat diselesaikan secara efektif tanpa mengorbankan kualitas fitur inti. Batasan masalah yang diterapkan antara lain:

- Aplikasi ditujukan khusus untuk sistem operasi macOS generasi terbaru dan tidak menyarar sistem operasi lain.
- Aplikasi berfokus pada audit konfigurasi dan deteksi anomali, bukan pada fungsi penanggulangan otomatis atau antivirus penuh.
- Deteksi anomali dibangun di atas data metrik jaringan teragregasi; aplikasi tidak melakukan analisis *payload* secara rinci.
- Penggunaan fitur pemindaian jaringan dan port terbatas pada jaringan yang dimiliki atau dikelola pengguna secara sah.

## F. Ruang Lingkup Laporan

Bagian ini mendefinisikan batas-batas materi yang akan dibahas dalam dokumen laporan ini, memastikan pembaca memiliki ekspektasi yang jelas mengenai konten dan analisis yang disajikan. Secara spesifik, laporan ini mencakup:

- **Pembahasan Aplikasi Berbasis Audit dan Deteksi Anomali**

Fokus utama laporan adalah pada mekanisme kerja modul Audit Keamanan OS (berbasis SystemCheck Swift) dan modul Deteksi Anomali Jaringan (berbasis *Python ML pipeline*).

- **Proses Desain, Implementasi, dan Hasil Pengujian Fitur**

Laporan menyajikan detail dari fase perancangan arsitektur (Engine/ dan Support/), implementasi fitur-fitur utama (termasuk integrasi *Python Bridge*), serta hasil pengujian fitur audit dan deteksi anomali.

- **Tidak Membahas Serangan Aktif atau *Exploit***

Laporan ini berfokus pada sisi defensif (audit konfigurasi) dan deteksi dini, dan tidak mendalami teknik *hacking* atau *exploit* perangkat lunak secara aktif.

- **Tidak Mencakup Analisis Keamanan Jaringan Tingkat *Enterprise***

Pembahasan analisis jaringan dan anomali terbatas pada lalu lintas dan konfigurasi pada perangkat macOS pengguna tunggal dan jaringan lokal (*on-device* dan *on-intranet*).

## G. Metode Pengembangan

Metode pengembangan ini menjelaskan langkah-langkah sistematis yang ditempuh dalam merancang, membangun, dan memverifikasi aplikasi *Aman*. Pendekatan yang digunakan mencakup kombinasi penelitian teori dan implementasi praktis yang iteratif:

- **Studi Literatur tentang Audit Keamanan dan *Anomaly Detection***

Mengumpulkan dan menganalisis literatur terkait standar penguatan macOS (misalnya CIS Benchmark), arsitektur aplikasi audit, dan teknik *machine learning* untuk deteksi anomali jaringan (*legacy*, *seasonality*, *changepoint*, *multivariate*, *newtalker*).

- **Analisis Kebutuhan Fitur Audit dan ML**

Mengidentifikasi kebutuhan fungsional dan non-fungsional, termasuk jenis-jenis pemeriksaan keamanan yang harus diimplementasikan (sekitar 70 *check*) dan *data contract* untuk *Python pipeline*.

- **Perancangan Arsitektur Aplikasi dan *Pipeline Deteksi***

Merancang arsitektur aplikasi yang modular (SwiftUI, Audit Engine, Network Toolkit) dan *pipeline* analitik (JSON I/O, analyzer.py).

- **Implementasi menggunakan SwiftUI + Python ML Backend**

Melakukan implementasi utama menggunakan *framework* SwiftUI untuk antarmuka pengguna, Swift untuk logika *front-end* dan *Audit Engine*, serta Python untuk *backend* deteksi anomali.

- **Pengujian menggunakan Skenario *Trafik* dan Audit Sistem Nyata**

Melaksanakan pengujian unit (Swift dan Python,) dan pengujian terintegrasi menggunakan skenario *traffic* jaringan simulasi serta kondisi audit sistem operasi yang bervariasi.

## BAB II Landasan Teori

### A. Pengertian Aplikasi

**Aplikasi Aman** didefinisikan sebagai aplikasi desktop macOS yang beroperasi secara mandiri (*on-device*), dirancang khusus untuk memenuhi kebutuhan penguatan keamanan sistem. Aplikasi desktop dipilih karena kemampuannya untuk berinteraksi langsung dengan sistem operasi dan menyediakan kinerja yang cepat, serta menjaga privasi data pengguna sepenuhnya.

Aplikasi keamanan dapat diklasifikasikan berdasarkan fungsi utamanya. Aplikasi Aman mengintegrasikan tiga peran tersebut:

#### 1. Aplikasi Audit (*Auditor*)

Bertujuan untuk memeriksa konfigurasi, kebijakan, dan izin sistem terhadap standar keamanan yang ditetapkan. Aplikasi Aman menjalankan fungsi ini melalui **Audit Engine** dan subkelas SystemCheck yang melakukan verifikasi konfigurasi macOS.

#### 2. Aplikasi Pemindai (*Scanner*)

Bertujuan untuk mengidentifikasi keberadaan entitas (misalnya port terbuka, layanan, atau host lain) dalam lingkungan jaringan. Aplikasi Aman menjalankan fungsi ini melalui modul Network Mapping dan Network Security Toolkit.

#### 3. Aplikasi Pemantauan (*Monitoring*)

Bertujuan untuk mengamati perilaku sistem dan lalu lintas jaringan secara berkelanjutan untuk mendeteksi penyimpangan. Aplikasi Aman menjalankan fungsi ini melalui Analyzer Pipeline berbasis *Machine Learning*.

### B. Konsep Dasar Cybersecurity Assessment

Konsep dasar *Cybersecurity Assessment* dalam konteks aplikasi Aman berfokus pada dua pilar utama: *Audit Konfigurasi Statis* dan *Analisis Perilaku Dinamis*. *Audit Konfigurasi Statis* melibatkan pemeriksaan dan validasi pengaturan sistem penting, seperti status enkripsi disk, perlindungan integritas sistem, konfigurasi pengamanan aplikasi, dan mode keamanan tinggi, yang sebagian besar diturunkan dari standar industri seperti CIS Benchmarks. *Analisis Perilaku Dinamis* (*Behavioral Analysis*) melengkapi audit statis dengan menganalisis metrik jaringan *real-time* untuk mengidentifikasi penyimpangan yang tidak dapat dideteksi dari konfigurasi semata.

- **Audit Sistem Operasi (CIS Benchmark & CMMC)**

Proses audit pada aplikasi Aman dirancang untuk mencerminkan gaya *Center for Internet Security (CIS)* dan *Cybersecurity Maturity Model Certification(CMMC) Benchmark*, yaitu serangkaian rekomendasi yang diakui industri untuk konfigurasi

sistem yang aman. Pemeriksaan ini mencakup area seperti enkripsi disk (FileVault), perlindungan integritas sistem (SIP), dan seluruh *checkups*.

- **Hardening dan Verifikasi Konfigurasi**

*Hardening* adalah tindakan mengamankan sistem dengan mengurangi permukaan serangan. Aplikasi *Aman* mendukung proses ini dengan menyediakan verifikasi (SystemCheck) dan merekomendasikan langkah remediasi yang jelas.

- **Observasi Perilaku Jaringan**

Berbeda dengan inspeksi *packet* mendalam, aplikasi *Aman* melakukan observasi melalui pengumpulan metrik lalu lintas jaringan teragregasi (*bytes*, *packets*, *flows*). Observasi ini menjadi input utama bagi *ML pipeline* untuk memahami pola aktivitas jaringan normal.

- **Deteksi Penyimpangan (Anomaly-based Detection)**

Berdasarkan observasi, *assessment* berfokus pada deteksi perilaku yang menyimpang dari pola normal yang dipelajari. Metode ini penting untuk mengidentifikasi ancaman zero-day atau aktivitas mencurigakan yang tidak memiliki tanda tangan (*signature*) yang diketahui.

## C. Konsep Machine Learning dalam Keamanan Sistem

Aplikasi *Aman* memanfaatkan *Machine Learning* (ML) untuk melakukan deteksi anomali pada lalu lintas jaringan. Pendekatan ini menawarkan kemampuan untuk mengidentifikasi ancaman yang lebih canggih dibandingkan metode tradisional.

- **Perbedaan *Signature-based* dan *Behavior/Anomaly-based***

- **Basis Berbasis Tanda Tangan (Signature-based)**

Metode ini membandingkan aktivitas yang muncul dengan daftar pola ancaman yang sudah dikenal sebelumnya, misalnya sidik jari file, perintah tertentu, atau pola komunikasi spesifik. Pendekatan ini efektif untuk ancaman yang sudah pernah terjadi, tetapi tidak mampu mengenali ancaman baru atau varian yang belum tercatat.

- **Berbasis Perilaku (Anomaly-based)**

Berfokus pada mengenali penyimpangan dari pola aktivitas normal. Jika sistem biasanya memiliki tingkat lalu lintas tertentu, namun tiba-tiba meningkat drastis atau muncul jenis komunikasi baru, sistem akan memandangnya sebagai anomali.

*Aman* menggunakan pendekatan ini, sehingga lebih mampu mendeteksi ancaman baru, termasuk aktivitas yang belum memiliki tanda tangan, seperti serangan zero-day atau pola komunikasi yang tidak umum.

- **Konsep Baseline, Pola Musiman, Perubahan Mendadak**

*ML pipeline* dalam *analyzer.py* bekerja dengan memodelkan tiga konsep perilaku jaringan:

- **Baseline (Kebiasaan Umum)**  
Merupakan gambaran perilaku jaringan yang paling umum terjadi. Jika biasanya lalu lintas berada pada tingkat tertentu, baseline akan mengenali itu sebagai "normal".
  - **Pola Musiman (Seasonality)**  
Pola penggunaan yang berulang, misalnya perangkat yang lebih aktif pada jam kerja dibandingkan malam hari. Pola ini membantu membedakan lonjakan yang wajar dengan lonjakan yang tidak lazim.
  - **Perubahan Mendadak (Changepoint)**  
Perubahan signifikan yang terjadi secara tiba-tiba dan cukup lama sehingga tidak bisa dianggap sebagai fluktuasi biasa. Perubahan seperti ini bisa menandai adanya aktivitas baru yang perlu diperhatikan, misalnya proses sistem yang tiba-tiba mengirim data dalam jumlah besar.
  - **Analisis Gabungan Beberapa Faktor (Multivariate)**  
Sistem tidak hanya menilai satu jenis data (misalnya jumlah byte), tetapi melihat beberapa aspek sekaligus seperti jumlah paket, jumlah koneksi, dan besarnya trafik.  
Ketika beberapa indikator menunjukkan pola yang selaras dan tidak wajar, risiko dianggap lebih tinggi. Analisis ini membantu mengenali perilaku kompleks yang tidak terlihat jika hanya melihat satu nilai saja.
  - **Pengenalan Entitas Baru (New Talker)**  
Sistem memantau apakah perangkat, proses, atau tujuan koneksi baru muncul untuk pertama kalinya dalam periode tertentu.  
Kehadiran "pembicara baru" bisa menjadi tanda munculnya proses yang tidak pernah aktif sebelumnya atau koneksi ke tujuan yang tidak dikenal.
- **Perspektif Fungsional**  
Laporan ini berfokus pada hasil fungsional dari deteksi anomali (skor, kode alasan, diagnostik reasonCodes) dan bagaimana temuan tersebut membantu pengguna, alih-alih mendalamai teknik matematis yang digunakan (seperti algoritma z-score atau L2 norm).
  - **Peran Konfigurasi dan Kendali Sistem**  
Pipeline analitik Aman bekerja berdasarkan berkas konfigurasi internal yang menentukan urutan tahapan analisis, parameter, dan detektor mana yang aktif. Selain itu, aplikasi juga menyediakan mekanisme kendali untuk menyalakan atau mematikan jenis analisis tertentu, sehingga pengguna bisa menyesuaikan tingkat sensitivitas sesuai kebutuhan.

## D. Arsitektur Aplikasi

Arsitektur aplikasi *Aman* dirancang secara modular dan terbagi menjadi dua komponen fungsional utama yang saling terintegrasi:

## 1. Auditor Sistem dan *Front-end*

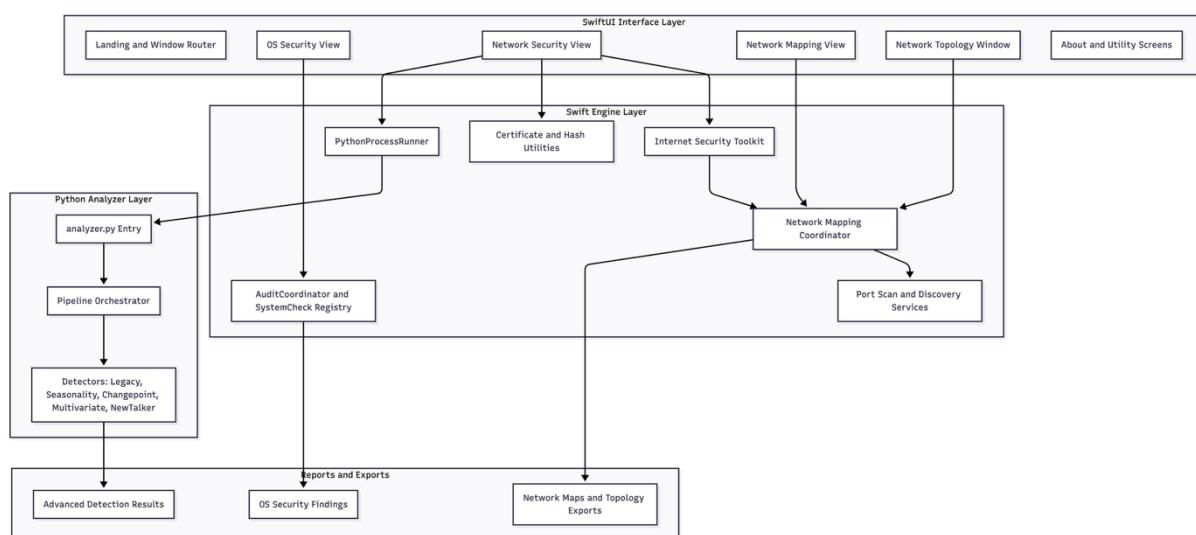
Komponen ini dibangun menggunakan SwiftUI dan Swift Engine. Fungsinya adalah menyediakan Antarmuka Pengguna (UI) dan menjalankan *Audit Engine* yang melakukan pemeriksaan konfigurasi sistem (SystemCheck).

## 2. Analyzer Jaringan ML *Backend*

Komponen ini berbasis Python (*analyzer.py*). Fungsinya adalah menerima metrik jaringan, menjalankan detektor anomali, dan menghasilkan output analitik.

Kedua komponen dihubungkan melalui *Python Bridge* (*PythonProcessRunner*). *Bridge* ini bertanggung jawab untuk memanggil skrip Python dan mengelola komunikasi data melalui I/O standar (JSON melalui *stdin* dan *stdout*).

Data dikumpulkan oleh *Swift Engine* dari sistem operasi dan jaringan lokal. Data ini dikonversi menjadi metrik terstruktur, yang kemudian diumpulkan ke *Python Analyzer*. Setelah diproses oleh model ML, hasil deteksi (skor anomali, kode alasan) dikirim kembali ke *Swift Engine* dan disajikan kepada pengguna dalam modul Network Analyzer.



Gambar 2 Diagram Arsitektur Aplikasi

## E. Etika Penggunaan dalam Peretasan Etis

Aplikasi *Aman* dikembangkan untuk mematuhi prinsip *white-hat* (peretasan etis) sepenuhnya, menjadikannya alat yang dirancang untuk memperkuat, bukan merusak. Prinsip etika yang mendasari penggunaan aplikasi ini antara lain:

- **Prinsip White-Hat: Tidak Merusak, Tidak Menyerang**

Aplikasi ini berfungsi sebagai alat audit dan deteksi defensif. Aplikasi ini tidak memiliki kemampuan untuk melakukan serangan aktif (*exploit*) atau manipulasi data yang merusak.

- **Menggunakan Aplikasi Hanya untuk Evaluasi Keamanan Milik Sendiri**

Penggunaan fitur pemetaan jaringan dan analisis anomali sangat dibatasi pada perangkat dan jaringan yang secara sah dimiliki atau berada di bawah otoritas pengguna. Intranet-sensitive checks diatur oleh mekanisme *consent store* (*NetworkWorkspaceConsentStore*).

- **Menghindari Pelanggaran Privasi dan Data**

Karena seluruh proses audit sistem dan analisis anomali berbasis ML berjalan secara lokal (*on-device*), aplikasi ini memastikan bahwa data sensitif pengguna (seperti metrik lalu lintas) tidak dikirim ke *cloud* atau pihak ketiga mana pun, sehingga menjaga privasi pengguna.

## BAB III Perancangan Sistem

### A. Gambaran Umum Sistem

Aplikasi *Aman* dirancang untuk beroperasi sebagai *dashboard* keamanan terintegrasi yang memungkinkan pengguna memverifikasi konfigurasi sistem mereka dan memantau perilaku jaringan secara bersamaan. Alur kerja keseluruhan aplikasi memisahkan tugas audit statis (konfigurasi sistem) dari analisis dinamis (perilaku jaringan), namun menyatukannya dalam satu *shell* pengalaman pengguna (SwiftUI Shell).

#### Alur Kerja Keseluruhan:

Aplikasi Aman mengintegrasikan tiga proses utama untuk memberikan gambaran keamanan yang holistik kepada pengguna macOS: Audit Statis (konfigurasi), Analisis Dinamis (perilaku jaringan), dan Pelaporan Terpadu.

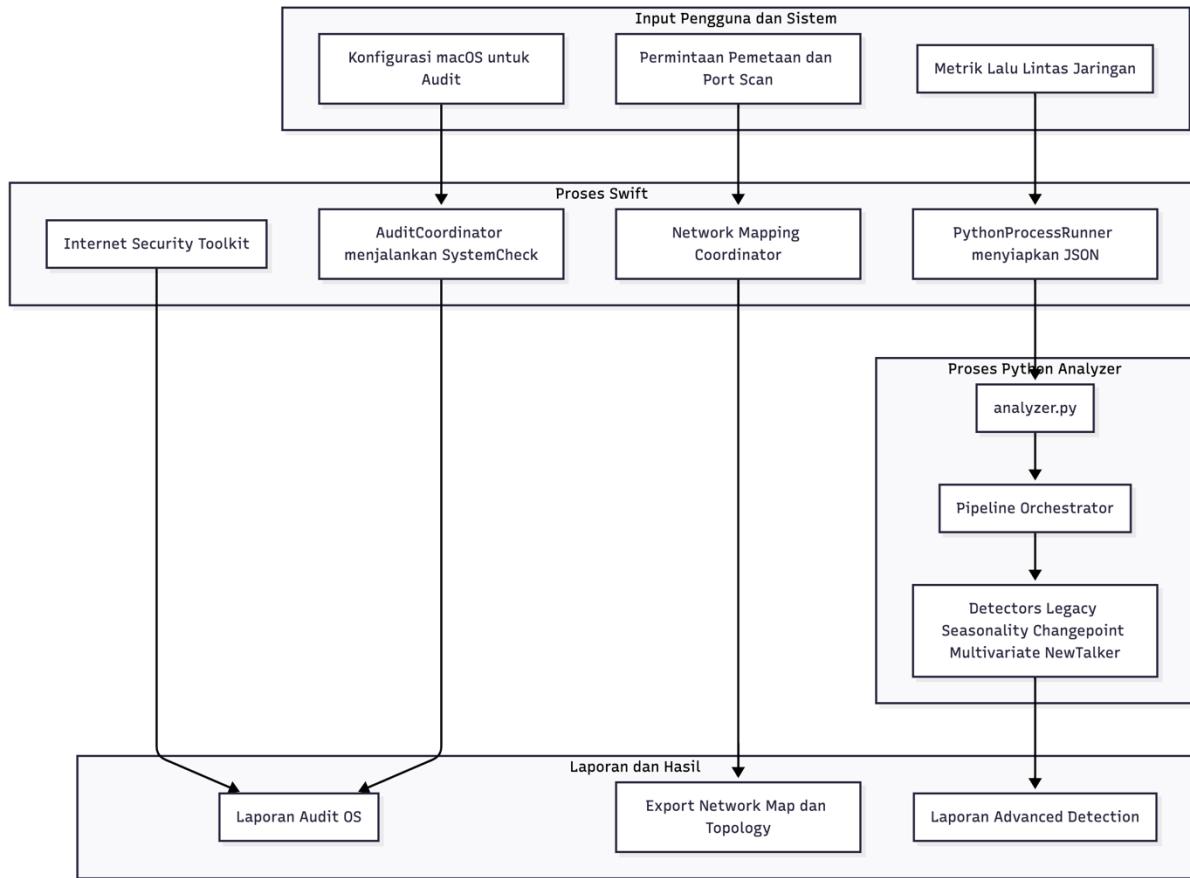
1. **Audit Keamanan OS (Audit Statis)** : Proses ini bertujuan menilai seberapa kuat konfigurasi bawaan macOS Anda.
  - **Aktivitas Pengguna:** Pengguna memilih modul **OS Security** dan memulai Run Audit.
  - **Fungsi Inti:** Audit Engine (inti aplikasi) menjalankan sekitar 70 pemeriksaan keamanan yang telah ditentukan. Pemeriksaan ini mencakup berbagai domain penting seperti enkripsi disk (FileVault), integritas sistem (SIP), dan kebijakan akun.
  - **Hasil:** Aplikasi menyajikan daftar temuan yang diklasifikasikan berdasarkan status dan tingkat keparahan (misalnya, *Low*, *Medium*, *High*, *Critical*). Setiap temuan disertai dengan langkah perbaikan (remediation) yang jelas.
2. **Analisis Jaringan dan Deteksi Anomali (Analisis Dinamis)** : Proses ini berfokus pada perilaku jaringan secara *real-time* untuk mengidentifikasi aktivitas mencurigakan yang tidak terdeteksi oleh audit statis.
  - **Pengumpulan Data:** Data metrik lalu lintas jaringan dikumpulkan secara lokal di perangkat. Fitur ini memerlukan persetujuan (consent) eksplisit dari pengguna sebelum berjalan.
  - **Proses Analitik:** Metrik waktu nyata (seperti *bytesPerSecond* atau *packetsPerSecond*) diteruskan ke Machine Learning Engine (pipeline Python) untuk diproses.
  - **Deteksi Lanjut:** Pipeline analitik menjalankan lima jenis detektor utama (termasuk detektor *changepoint*, *multivariate*, dan *newtalker*) untuk mengidentifikasi pola lalu lintas yang tidak lazim.

- **Hasil:** Aplikasi menyajikan skor anomali, kode alasan, dan diagnostik. Peringatan akan muncul di antarmuka jika skor risiko melewati ambang batas yang ditentukan.
3. **Visualisasi dan Laporan :** Semua temuan disajikan secara terpadu di antarmuka pengguna yang *native* dan responsif.
- **Visualisasi:** Hasil audit (temuan) dan skor anomali dari ML divisualisasikan secara *real-time*. Antarmuka mendukung pemfilteran temuan, pencarian, dan navigasi yang jelas.
  - **Pelaporan:** Pengguna dapat mengekspor hasil temuan audit OS ke dalam format **HTML**. Hasil pemetaan jaringan juga dapat diekspor dalam format lain (seperti JSON/graph).

#### **Hubungan Antar Modul:**

Integrasi yang erat antara audit OS, analisis jaringan, dan deteksi ML memungkinkan pengguna mendapatkan gambaran keamanan Holistik:

- **Audit OS:** Menjawab pertanyaan "Apa yang salah dengan **konfigurasi** saya?"
- **Network Analysis (Toolkit/Mapping):** Menjawab pertanyaan "Apa **paparan** jaringan saya?"
- **ML Detection:** Menjawab pertanyaan "Apakah ada **perilaku** mencurigakan saat ini?"



Gambar 3 Alur Kerja Keseluruhan Aplikasi Aman (Input -> Pemrosesan Swift/Python -> Output Laporan)

## B. Arsitektur Utama

Arsitektur aplikasi **Aman** dirancang agar modular dan terpisah secara logis, yang memungkinkan fleksibilitas dalam pengembangan dan pemeliharaan. Secara konseptual, aplikasi ini terbagi menjadi tiga komponen utama:

### 1. Lapisan Antarmuka Pengguna (UI Layer)

Ini adalah lapisan yang berinteraksi langsung dengan pengguna dan bertanggung jawab atas tampilan serta navigasi.

- **Teknologi:** Dibangun menggunakan SwiftUI.
- **Fungsi:**
  - Bertindak sebagai *shell* utama yang mengelola navigasi antar jendela utama, seperti OS Security, Network Security, Network Topology, dan About.
  - Menyajikan hasil audit, skor anomali, dan topologi jaringan kepada pengguna.

- UI terikat secara reaktif (*binds*) ke ObservableObject view models (misalnya AuditCoordinator dan NetworkMappingCoordinator) yang mengorkestrasi semua tugas asinkron, memastikan antarmuka tetap responsif.

## 2. Audit dan Kontrol Mesin (Audit Engine / Swift Core)

Ini adalah lapisan logika bisnis inti yang ditulis dalam Swift.

- **Teknologi:** Core Logic utama berada di folder Engine/ dan Modules/.
- **Fungsi:**
  - **Audit Engine**  
Bertanggung jawab untuk menjalankan semua pemeriksaan keamanan (SystemCheck) yang mendefinisikan audit macOS. Hasil pemeriksaan dicatat sebagai AuditFinding.
  - **Network Toolkit & Mapping**  
Menyediakan logika untuk fitur utilitas seperti *Certificate Lookup* (klien crt.sh), *Hash Generator*, dan mengelola proses pemetaan jaringan serta pemindaian port.
  - **Python Bridge (PythonProcessRunner)**  
Bertindak sebagai perantara komunikasi. Komponen ini bertanggung jawab untuk meluncurkan ML Engine (*analyzer.py*), menulis data input (JSON Request) ke *stdin*, dan membaca hasilnya (JSON Output) dari *stdout*.

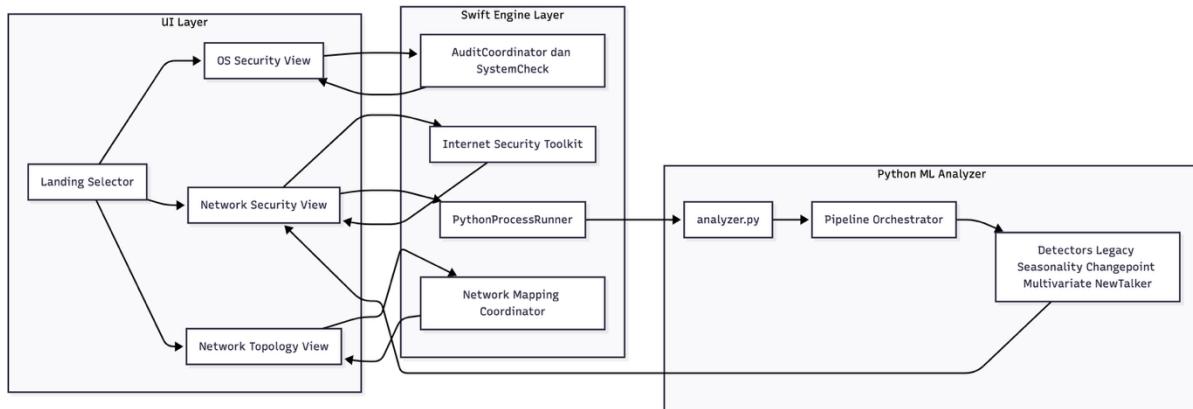
## 3. Mesin Analisis ML (ML Engine / Python Backend)

Ini adalah lapisan analitik yang menangani pemrosesan data jaringan yang kompleks.

- **Teknologi:** Pipeline analitik sepenuhnya berbasis Python. *Entry point* utama adalah skrip analyzer.py yang terletak di folder Support/.
- **Fungsi:**
  - Menerima data metrik jaringan (misalnya, *bytesPerSecond*, *packetsPerSecond*) yang dikirimkan oleh Audit Engine.
  - Membangun dan menjalankan pipeline deteksi anomali menggunakan manifes konfigurasi (yaml).
  - Mengeksekusi serangkaian detektor (*legacy*, *seasonality*, *changepoint*, *multivariate*, *newtalker*) secara berurutan.
  - Menghitung skor anomali gabungan dan menghasilkan diagnostik serta *reason codes* sebelum mengirimkan hasilnya kembali ke Audit Engine.

## Perpindahan Data Antar Modul

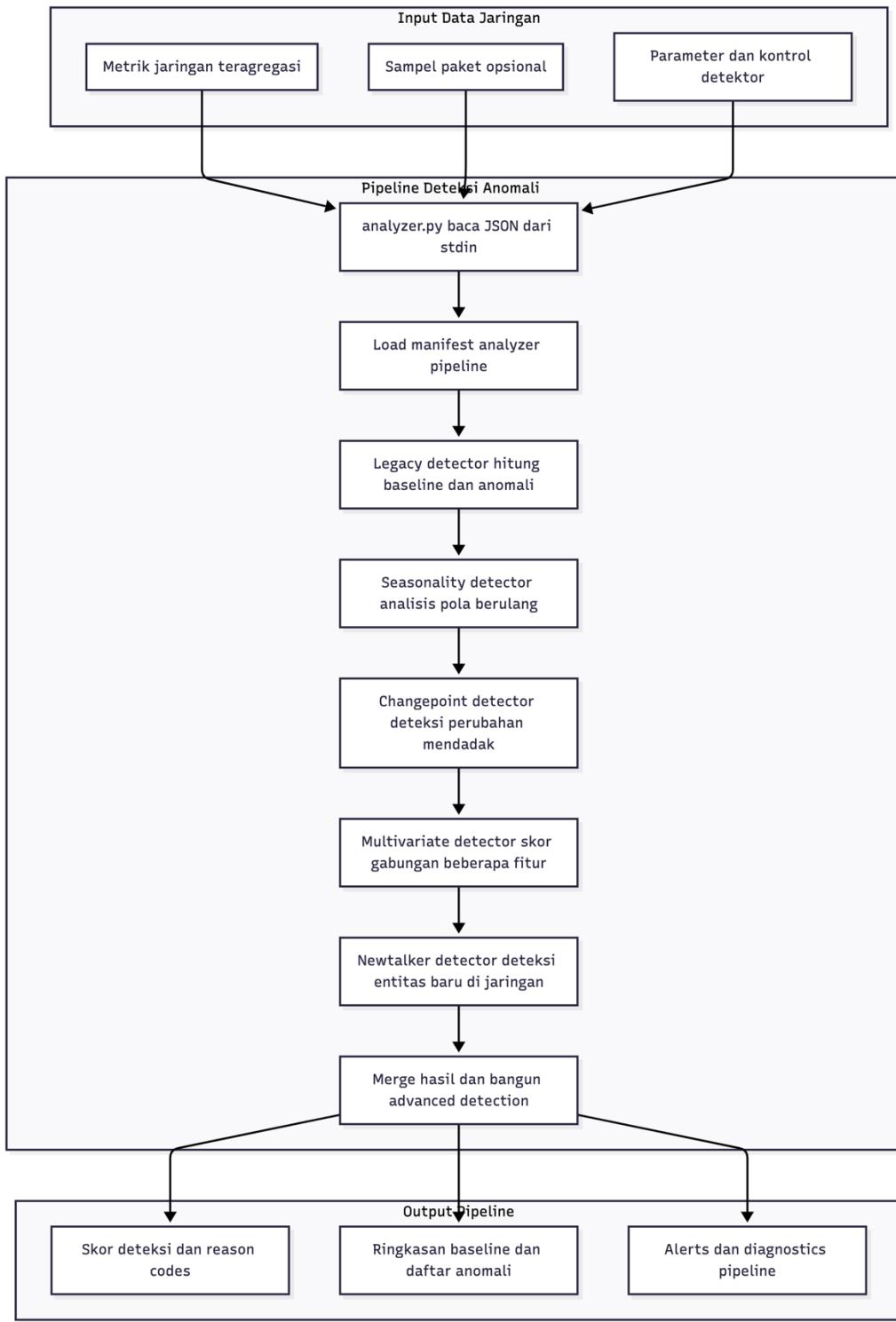
Data bermula dari sistem/jaringan (dikumpulkan oleh *Swift Engine*). Data diubah menjadi JSON Request (termasuk *metrics* dan *params*) dan dikirimkan ke Python Bridge. *ML Engine* memproses data dan mengembalikan JSON Output (*advancedDetection*) ke *Swift Engine* untuk divisualisasikan di UI.



Gambar 4 Diagram Arsitektur Tiga Lapis Aplikasi Aman

## C. Diagram Alur Proses Deteksi Anomali

Proses inti deteksi anomali jaringan dikelola oleh *Python-only analytics pipeline*. Proses ini bersifat sekuensial dan modular, memungkinkan konfigurasi ulang detektor melalui file manifest.



Gambar 5 Flowchart Pipeline Deteksi Anomali ML

### Penjelasan Tahap Pipeline Deteksi Anomali:

#### 1. Input Data

Pipeline menerima data metrik lalu lintas jaringan yang telah diringkas, seperti jumlah byte per detik, jumlah paket, jumlah aliran koneksi, serta informasi tambahan bila tersedia. Data dikirim dalam bentuk JSON sehingga setiap tahap dapat membaca nilai yang konsisten. Pengguna (atau aplikasi) juga dapat memberikan parameter tambahan untuk menyesuaikan sensitivitas analisis.

## **2. Orkestrasi Pipeline**

Sebelum analisis dimulai, sistem memuat sebuah berkas konfigurasi yang menentukan detektor mana saja yang dipakai, urutan pemrosesan, dan aturan dasarnya. Konfigurasi ini membuat pipeline fleksibel: detektor tertentu bisa dimatikan, urutan bisa diubah, dan batasan tertentu bisa diatur tanpa mengubah kode.

## **3. Eksekusi Detektor**

Setiap detektor memiliki fokus analisis yang berbeda. Seluruh detektor dijalankan satu per satu, sehingga hasil detektor sebelumnya dapat memperkaya detektor berikutnya.

### **a. Legacy**

Menentukan gambaran dasar (baseline) perilaku jaringan, kemudian mencari penyimpangan dari kebiasaan tersebut. Tahap ini membantu mengenali anomali sederhana yang terlihat dari kenaikan mendadak atau pola yang berubah drastis.

### **b. Seasonality**

Mendeteksi pola yang berulang, seperti penggunaan tinggi pada jam kerja atau penurunan lalu lintas di malam hari. Jika lalu lintas menyimpang dari pola rutin, maka dianggap tidak wajar.

### **c. Changepoint**

Mencari perubahan mendadak yang bertahan lama. Tahap ini baik untuk mengenali kejadian besar, misalnya munculnya proses baru yang tiba-tiba mengirim banyak data.

### **d. Multivariate**

Melihat beberapa metrik sekaligus untuk mencari hubungan antar variabel. Detektor ini membantu mendeteksi pola kompleks yang tidak terlihat jika hanya melihat satu jenis data.

### **e. Newtalker**

Menyelidiki apakah ada tujuan, proses, atau port baru yang tiba-tiba aktif dalam

jaringan. Kemunculan entitas baru sering menjadi tanda komunikasi asing yang perlu diperhatikan.

#### 4. Penggabungan Hasil

Semua hasil dari lima detektor digabungkan menjadi satu konteks bersama.

Penggabungan ini mencakup:

- skor tingkat kejanggalan,
- alasan mengapa sebuah aktivitas dinilai anomali,
- perubahan atau pola yang teridentifikasi,
- informasi pendukung lainnya.

Tujuannya: memberikan gambaran menyeluruh, tidak hanya potongan-potongan hasil terpisah.

#### 5. Output Hasil

Pipeline menghasilkan keluaran dalam bentuk JSON yang memuat:

- Advanced Detection (skor risiko final dan alasan pendukung),
- Ringkasan perilaku jaringan,
- Informasi tambahan seperti perubahan pola, entitas baru, atau anomali kompleks,
- Peringatan (alert) apabila skor melebihi batas tertentu.

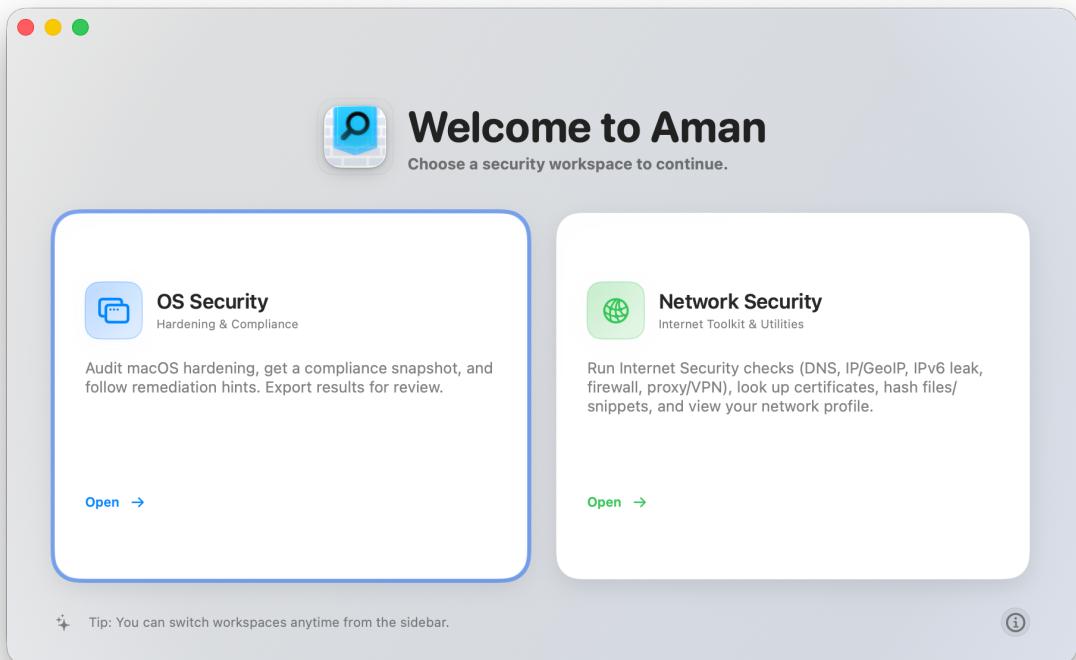
Hasil inilah yang diteruskan kembali ke aplikasi untuk ditampilkan dalam antarmuka pengguna.

### D. Desain Antarmuka Pengguna

Desain UI menggunakan SwiftUI (Apple Human Interface Guidelines) untuk memastikan pengalaman pengguna yang intuitif dan *native* pada macOS.

#### 1. Layout Utama Aplikasi

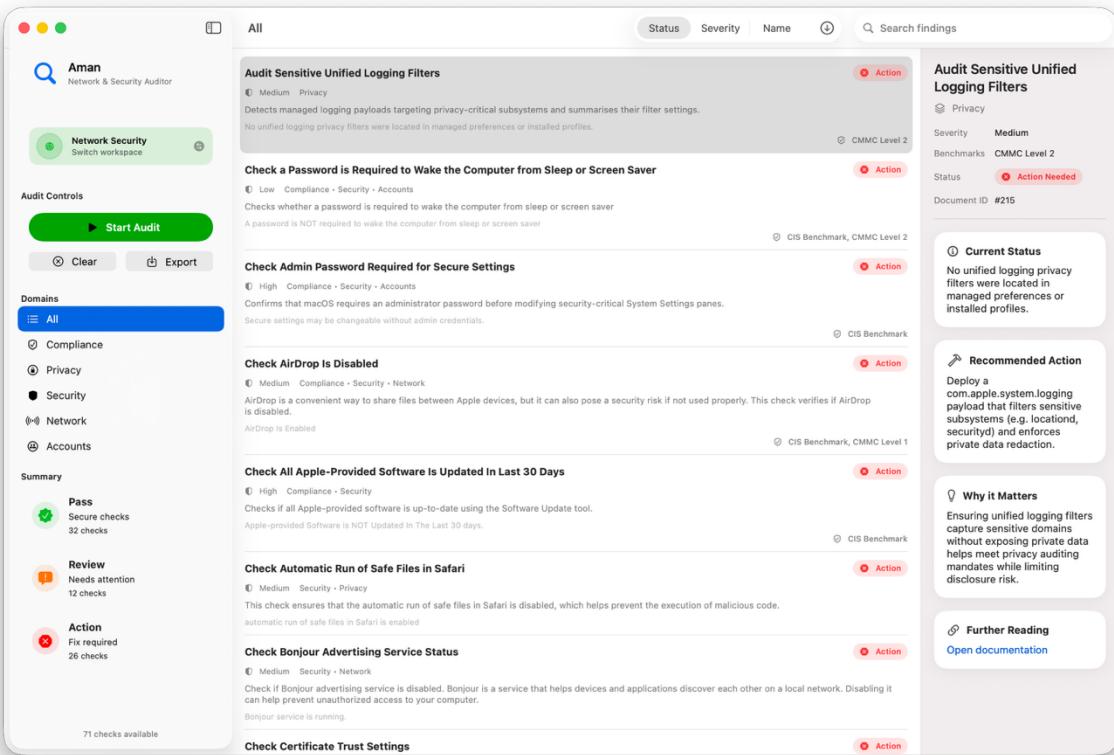
- **Landing Selector:** Setelah diluncurkan, pengguna disambut dengan jendela pemilihan untuk masuk ke modul utama: OS Security, Network Security, Network Topology, dan About.
- **Navigation:** Jendela dapat dibuka kembali melalui menu *Window* atau tombol *toolbar*.



Gambar 6 Tampilan Landing Selector Utama Aplikasi

## 2. Tampilan Audit Keamanan OS

- **Daftar Temuan:** Menampilkan temuan audit dalam format daftar yang dapat dicari (searchable), difilter (berdasarkan domain), dan diurutkan (berdasarkan Status, Severity, atau Nama).
- **Detail Temuan:** Saat sebuah baris dipilih, jendela detail menampilkan title, description, remediation yang jelas dan dapat ditindaklanjuti, severity (Low/Medium/High/Critical/Info), dan referensi dokumentasi.
- **Aksi Cepat:** Terdapat tombol untuk Run Audit dan tombol HTML Export untuk menyimpan hasil audit.



Gambar 7 Tampilan Modul OS Security dengan Daftar Temuan dan Detail Remediasi

### 3. Tampilan Analisis Jaringan (Network Analyzer)

- **Visualisasi Data**

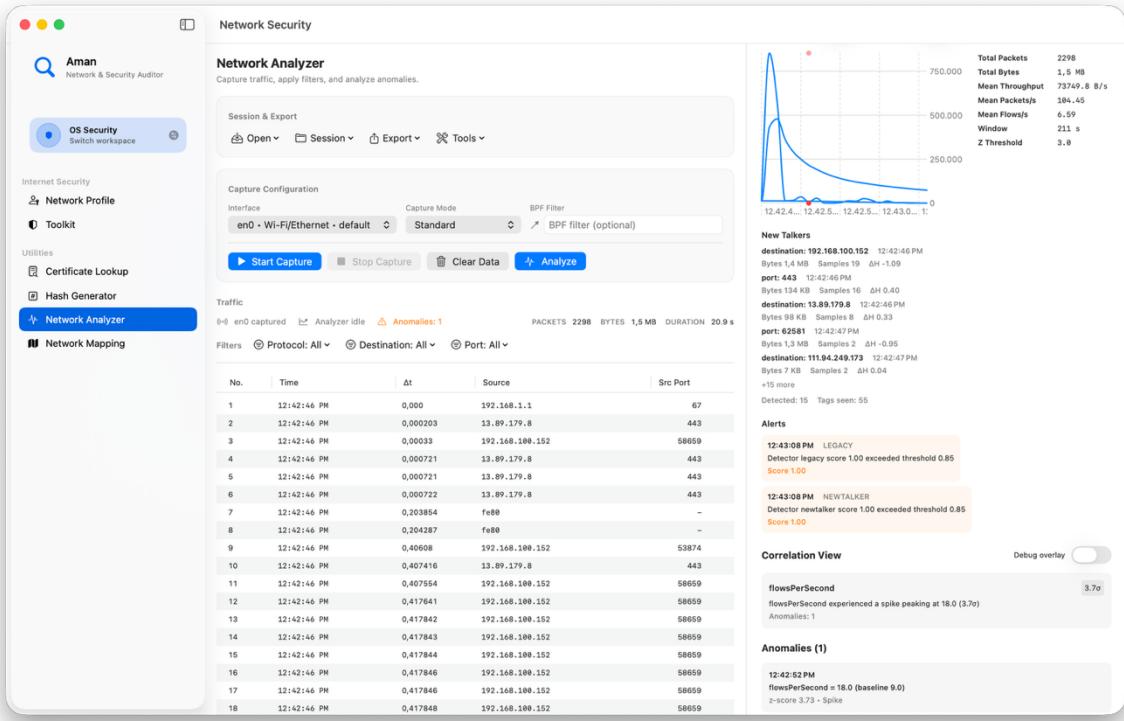
Modul ini menampilkan visualisasi grafik (*graph rendering*) data metrik jaringan menggunakan *WindowView*.

- **Hasil Deteksi**

Skor anomali yang dihasilkan oleh *ML Engine* direfleksikan dalam Advanced Detection Views (skor, *reason codes*, dan diagnostik). Peringatan muncul ketika skor melebihi ambang batas.

- **Network Security Suite**

Antarmuka dengan bilah sisi (*sidebar*) yang mengelompokkan Toolkit, Certificate Lookup, Hash Generator, dan Network Mapping.



Gambar 8 Tampilan Modul Network Analyzer dengan Grafik Metrik dan Skor Anomali

## E. Deskripsi Fitur-Fitur Utama

Aplikasi *Aman* memiliki serangkaian fitur inti yang terbagi dalam tiga domain utama:

### 1. Audit Keamanan OS:

a. **Sistem Cek Komprehensif:** Sekitar 70 cek (SystemCheck) yang mencakup aspek EFI, FileVault, SIP, Gatekeeper, *Updates*, *Sharing*, *Bluetooth/AirDrop*, dan kebijakan kata sandi. Aplikasi menjalankan audit konfigurasi yang komprehensif, dikategorikan ke dalam modul-modul berikut:

#### 1. Sistem Integrity dan Firmware

Memastikan perlindungan Secure Enclave, status SIP, dan enkripsi penuh disk (FileVault).

#### 2. Access Control dan Authentication

Menegakkan penggunaan kredensial, kebijakan kata sandi yang memadai, dan otentikasi setelah bangun dari mode tidur.

#### 3. Network dan Remote Access

Memeriksa status firewall, meninjau akses jarak jauh (SSH), membatasi layanan berbagi yang tidak perlu, dan melakukan pemindaian port internal.

#### 4. Software Update dan Patch Management

Memastikan pembaruan sistem dan keamanan berjalan otomatis, serta memverifikasi status Rapid Security Response (RSR).

#### 5. **System and Service Configuration**

Memastikan Gatekeeper aktif, menilai penerapan sandbox, dan meninjau Lockdown Mode.

#### 6. **Privacy and Telemetry**

Meninjau pengaturan pengiriman data diagnostik dan pengaturan pelacakan/personalisasi iklan.

#### 7. **Media and Peripheral Control**

Mengendalikan visibilitas dan izin berbagi melalui AirDrop, AirPlay, dan pengaturan Bluetooth.

#### 8. **Backup, Time, and Miscellaneous**

Memastikan cadangan otomatis terenkripsi dan sinkronisasi waktu terjaga.

#### 9. **Malware and Threat Protection**

Memeriksa mekanisme perlindungan malware bawaan dan menilai pengaturan browser terkait risiko infeksi.

### b. **Eksport HTML**

Fitur untuk mengekspor temuan audit ke format HTML untuk pelaporan dan dokumentasi.

## 2. Keamanan dan Pemetaan Jaringan:

### a. **Network Toolkit**

Alat untuk memverifikasi keamanan internet dasar seperti *DNS leak*, *IP exposure*, *firewall status*, dan indikator proxy/VPN.

### b. **Network Mapping & Topology**

Kemampuan untuk menemukan *host* pada jaringan lokal, menjalankan pemindaian *port* bertarget, dan memvisualisasikan koneksi jaringan (*Network Topology window*).

### c. **Certificate Lookup**

Utilitas untuk mengkueri *crt.sh* (Certificate Transparency logs) untuk domain tertentu.

## 3. Deteksi Anomali Jaringan (ML):

### a. **Pipeline Analitik On-Device**

Menggunakan *Python pipeline* dengan detektor legacy, seasonality, changepoint, multivariate, dan newtalker untuk analisis perilaku jaringan.

### b. **Persetujuan Data**

Fitur-fitur yang melibatkan analisis lalu lintas sensitif diatur oleh penyimpanan konsen (*NetworkWorkspaceConsentStore*).

# BAB IV Implementasi Aplikasi dan Hasil

Bab ini menguraikan lingkungan teknis, implementasi fungsionalitas utama, integrasi antarmuka pengguna, alur penggunaan, serta hasil akhir pengembangan aplikasi Aman.

## A. Lingkungan Pengembangan

Implementasi aplikasi Aman didedikasikan untuk platform macOS, yang memerlukan lingkungan pengembangan spesifik:

- **Sistem Operasi Target & IDE**

Aplikasi ini ditujukan untuk macOS Tahoe (26) atau versi yang lebih baru dan dikembangkan menggunakan Xcode 16 (atau yang lebih baru) untuk membangun dan mengelola proyek SwiftUI/Swift.

- **Bahasa Pemrograman:**

- **Swift/SwiftUI:** Digunakan sebagai bahasa utama untuk membangun Antarmuka Pengguna (UI), Audit Engine, dan seluruh logika kontrol aplikasi.
- **Python 3.9+:** Digunakan untuk Backend Analitik yang menjalankan proses deteksi anomali berbasis Machine Learning.

- **Ketergantungan (Dependencies)**

Aplikasi memanfaatkan System APIs macOS untuk pemeriksaan audit konfigurasi, serta membutuhkan akses internet untuk layanan seperti Certificate Lookup (melalui crt.sh).

## B. Implementasi Fitur-Fitur Aplikasi

Fitur utama aplikasi diimplementasikan dengan memisahkan logika inti dari presentasi, yang terbagi dalam tiga domain: Audit Sistem, Analisis Jaringan, dan Deteksi Anomali ML.

### 1. Implementasi Audit Keamanan OS

Proses audit keamanan macOS diwujudkan melalui serangkaian pemeriksaan terstruktur:

- **Anatomii Pemeriksaan (Check)**

Setiap pemeriksaan keamanan diimplementasikan sebagai unit mandiri yang disebut SystemCheck. Setiap unit diinisialisasi dengan metadata lengkap, termasuk judul, deskripsi, langkah remediasi yang jelas, tingkat keparahan (Low/Medium/High/Critical/Info), dan ID Dokumen unik (docID) untuk referensi.

- **Logika Eksekusi**

Logika audit (misalnya memeriksa status FileVault, konfigurasi Firewall, atau kebijakan kata sandi) dieksekusi secara asinkron. Hasilnya kemudian menentukan checkstatus (Green untuk Lulus, Yellow untuk Peringatan/Tinjau, Red untuk Tindakan) dan pesan status yang dapat dibaca manusia.

- **Orkestrasi**

Semua pemeriksaan didaftarkan dan dikelola oleh AuditCoordinator. Koordinator ini menjalankan pemeriksaan secara berurutan atau paralel, mengaplikasikan kategori fungsional (seperti *privacySuite* atau *networkingSuite*), dan secara otomatis memberikan tag *benchmark* (misalnya CMMC) untuk memudahkan pelaporan.

## 2. Implementasi Pipeline Deteksi Anomali ML

Deteksi anomali jaringan pada aplikasi Aman berjalan sepenuhnya di perangkat pengguna. Analisis dilakukan oleh mesin Python yang menerima data dari logika Swift melalui mekanisme pertukaran data sederhana namun terstruktur.

- **Pusat Analitik**

Mesin analitik (analyzer.py) berfungsi sebagai titik masuk utama. Mesin ini menerima data metrik jaringan yang sudah diringkas oleh aplikasi Swift dalam format JSON melalui standard input. Setelah diproses, mesin mengembalikan hasil analisis berupa skor anomali dan ringkasan perilaku jaringan melalui standard output. Dengan pendekatan ini, seluruh proses tetap lokal tanpa mengirimkan data ke server eksternal.

- **Orkestrasi Pipeline**

Analisis dijalankan melalui sebuah pipeline yang diatur oleh berkas konfigurasi YAML (analyzer\_pipeline.yaml). Berkas ini menetapkan urutan detektor, parameter yang digunakan, serta jenis analisis yang diaktifkan. Pipeline dapat diperluas atau diubah tanpa mengubah kode utama, sehingga tetap fleksibel.

- **Detektor Inti**

Pipeline menggunakan lima detektor yang masing-masing memiliki peran berbeda dalam memahami perilaku jaringan:

1. Legacy, mengidentifikasi penyimpangan dari kebiasaan umum (baseline) menggunakan pendekatan statistik sederhana.

2. Seasonality, mendeteksi pola penggunaan yang berulang dan menilai apakah suatu kenaikan atau penurunan lalu lintas masih berada dalam pola wajar.
3. ChangePoint, mengenali perubahan mendadak yang bertahan cukup lama, misalnya lonjakan lalu lintas dari proses tertentu.
4. Multivariate, menilai perilaku jaringan dari beberapa sudut sekaligus, seperti byte, paket, dan jumlah aliran, untuk menemukan pola anomali yang lebih kompleks.
5. NewTalker, menemukan entitas baru di jaringan, seperti host, port, atau proses yang baru pertama kali muncul dalam periode analisis.
6. Kelima detektor ini saling melengkapi sehingga pipeline dapat menangkap berbagai bentuk ketidakwajaran, dari perubahan sederhana hingga pola komunikasi yang benar-benar baru.

- **Hasil Keluaran**

Pipeline menghasilkan sebuah struktur JSON yang memuat advancedDetection, yaitu hasil gabungan dari semua detektor. Struktur ini mencakup skor risiko, alasan deteksi (reason codes), serta informasi diagnostik yang menyederhanakan interpretasi hasil bagi pengguna non-ahli. Selain itu, ringkasan perilaku seperti anomali yang ditemukan, perubahan pola, atau entitas baru juga disertakan agar pengguna dapat memahami konteksnya.

## C. Integrasi Antarmuka Pengguna

Integrasi dirancang untuk responsif, mengutamakan privasi, dan memberikan visualisasi yang efektif.

### 1. Desain Antarmuka (SwiftUI)

- **Struktur UI**

Antarmuka (ditempatkan di folder `view/`) dibangun dengan SwiftUI dan dipandu oleh `AmanApp.swift` untuk *window routing* (seperti ke *OS Security*, *Network Topology*, *About*).

- **Data Binding Reaktif**

Aplikasi menggunakan pola `ObservableObject` (`view models` seperti `AuditCoordinator` dan `NetworkSecurityViewModel`) untuk mengikat data secara reaktif ke UI. Ini berarti setiap pembaruan hasil audit, pemetaan jaringan, atau skor anomali secara otomatis tercermin di layar tanpa perlu *refresh* manual.

### 2. Python Bridge

Komunikasi antara logika Swift dan mesin analitik Python merupakan aspek krusial:

- **PythonProcessRunner**

Komponen ini berada di lapisan logika inti dan bertindak sebagai jembatan. Tugasnya adalah meluncurkan Python sebagai subproses, memformat data metrik dari Swift menjadi JSON Request, dan membacanya kembali.

- **Manajemen Status dan Kegagalan**

Bridge ini secara aktif memonitor status subproses, menangani timeout (dibatasi 15 detik), *crash*, dan skenario di mana Python tidak ditemukan, lalu melaporkan kegagalan tersebut kembali ke antarmuka pengguna untuk tindakan korektif.

- **Persetujuan Pengguna**

Fitur yang melibatkan pemindaian jaringan (*intranet scans*) atau analisis paket secara eksplisit meminta persetujuan pengguna terlebih dahulu dan menyimpannya melalui NetworkWorkspaceConsentStore untuk menjaga kepatuhan etis dan privasi.

## D. Alur Penggunaan Aplikasi oleh Pengguna

Alur kerja aplikasi dirancang untuk mengarahkan pengguna secara intuitif dari audit konfigurasi ke pemantauan perilaku jaringan:

1. **Peluncuran dan Navigasi**

Pengguna membuka aplikasi dan memilih modul melalui *landing selector* (misalnya OS Security atau Network Security).

2. **Menjalankan Audit OS**

Pengguna mengklik Run Audit di modul OS Security. Hasil dari sekitar 70 pemeriksaan disajikan dalam daftar yang dapat dicari dan disortir. Pengguna dapat memilih temuan untuk melihat detail remediation dan mengekspor hasilnya ke format HTML untuk pelaporan.

3. **Menggunakan Network Security Suite**

Pengguna beralih ke modul Network Security untuk mengakses berbagai alat:

- a. **Toolkit Internet:** Untuk pemeriksaan kebocoran DNS, paparan IP, atau status firewall.
- b. **Network Mapping:** Untuk menemukan *host* dan layanan di jaringan lokal, serta melakukan pemindaian *port*.
- c. **Network Analyzer:** Untuk mengaktifkan modul deteksi anomali.

4. **Alur Deteksi Anomali (Analyzer Flow)**

Metrik lalu lintas dikumpulkan dan diteruskan ke *Python pipeline*. Skor anomali yang dihasilkan ditampilkan. Peringatan dimunculkan jika skor risiko melebihi ambang batas yang dikonfigurasi, didukung oleh *reason codes* dari multi-detektor.

5. **Pelaporan dan Troubleshooting**

Hasil audit, pemetaan, dan analitik dapat diekspor. Aplikasi juga menyediakan log diagnostik di `~/Library/Logs/Aman/analyzer-last.json` untuk membantu pengguna jika terjadi kegagalan pemrosesan analitik.

## E. Keunggulan dan Kelemahan Aplikasi

### Keunggulan Aplikasi

- **Operasi Lokal (Privasi)**

Seluruh proses analisis utama (audit sistem dan ML *pipeline*) dilakukan secara lokal (on-device), memastikan data sensitif pengguna tidak dikirim ke layanan eksternal.

- **Integrasi Holistik**

Aplikasi mengintegrasikan audit konfigurasi sistem, pemetaan jaringan, dan deteksi anomali berbasis ML dalam satu platform, memberikan gambaran keamanan yang menyeluruh.

- **Desain Modular**

Arsitektur modular mempermudah penambahan pemeriksaan (SystemCheck) dan teknik analitik baru (*Python detectors*) di masa mendatang, mendukung pengembangan berkelanjutan.

- **Antarmuka Pengguna Native**

Menggunakan SwiftUI untuk menyediakan antarmuka yang *native*, responsif, dan mudah digunakan pada platform macOS.

### Kelemahan Aplikasi

- **Ketergantungan Platform**

Aplikasi sangat bergantung pada macOS Tahoe (26) atau yang lebih baru dan Python 3.9+, sehingga sangat tidak *portable* ke platform lain.

- **Hasil Deteksi Probabilistik**

Hasil deteksi anomali berbasis ML bersifat probabilistik dan dapat menghasilkan *false positive* (peringatan palsu) atau *false negative* (ancaman yang terlewatkan), meskipun telah diupayakan mitigasi melalui penggabungan skor dari multi-detektor.

- **Cakupan Fungsional Terbatas**

Aplikasi tidak mengantikan kebutuhan akan antivirus penuh, fungsi penanggulangan otomatis, atau kebijakan keamanan organisasi yang komprehensif. Fokusnya terbatas pada audit konfigurasi dan deteksi anomali.

## BAB V Penutup

### A. Kesimpulan

Pengembangan aplikasi *Aman* telah berhasil mengintegrasikan audit keamanan sistem macOS dan deteksi anomali jaringan berbasis *Machine Learning* dalam satu platform yang mengutamakan privasi dan beroperasi secara lokal (*on-device*).

Beberapa poin kunci yang dapat disimpulkan meliputi:

- **Fungsi Holistik**

Aplikasi ini sukses mengawinkan audit konfigurasi statis dengan analisis perilaku dinamis, memberikan landasan yang kuat bagi pengguna untuk memahami dan meningkatkan keamanan perangkat macOS mereka.

- **Keunggulan Privasi**

Seluruh proses analisis utama berjalan secara lokal, menjamin bahwa data sensitif pengguna tidak dikirim ke layanan eksternal.

- **Deteksi Tingkat Lanjut**

Implementasi *pipeline* ML dengan lima detektor utama memungkinkan identifikasi pola lalu lintas tidak biasa yang tidak dapat dideteksi oleh konfigurasi statis semata.

- **Orientasi Pengguna**

Aplikasi ini menyajikan temuan audit dan skor anomali dalam bentuk visual dan naratif yang mudah dipahami, lengkap dengan rekomendasi perbaikan (*remediation*) yang jelas.

### B. Saran Pengembangan

Untuk meningkatkan nilai dan kapabilitas aplikasi *Aman* di masa mendatang, beberapa saran pengembangan yang dapat dipertimbangkan adalah:

- **Memperluas Cakupan Pemeriksaan Keamanan**

Memperluas cakupan pemeriksaan keamanan dan menyesuaikannya dengan standar atau regulasi baru yang muncul, serta menambahkan integrasi *benchmark* yang lebih mendalam (misalnya tag CMMC).

- **Penyempurnaan Metode Deteksi Anomali**

Menyempurnakan metode deteksi anomali dengan teknik yang lebih canggih dan adaptif terhadap perubahan pola penggunaan, guna mengurangi *false positive* dan *false negative*. Hal ini dapat mencakup *tuning* parameter detektor atau penambahan model *clustering* untuk *anomaly narrative* yang lebih kuat.

- **Peningkatan Kemampuan Visualisasi**

Meningkatkan kemampuan visualisasi, khususnya untuk topologi jaringan dan *timeline* anomali, sehingga pengguna dapat melacak insiden secara kronologis.

- **Integrasi Eksternal**

Menyediakan opsi integrasi dengan alat pemantauan atau pelaporan eksternal bagi organisasi yang membutuhkan, sambil tetap memastikan aspek privasi menjadi opsi yang dikontrol penuh oleh pengguna.

## Daftar Pustaka

1. **Apple Developer Documentation.** (2025). *SwiftUI: Declare the user interface and behavior for your app on every platform.* Apple Inc. [Apple Developer Documentation - SwiftUI. Diakses **20 November 2025**].
2. **Center for Internet Security (CIS).** (2025). *CIS Benchmark for Apple macOS 26 Tahoe (Versi 1.0.0).* [CIS Benchmarks List - CIS Center for Internet Security. Diakses **20 November 2025**].
3. **Mohammed, R., & Akay, M. F.** (2023). Anomaly Detection in Network Traffic Using Machine Learning. *Cukurova University Journal of Natural & Applied Sciences*, 2(3), 5-12.
4. **Murshed, A. M. B., et al.** (2025). *dtaianomaly* A Python library for time series anomaly detection. [arXiv preprint: 2502.14381].
5. **Patel, D., Srinivasan, K., Chang, C. Y., Gupta, T., & Kataria, A.** (2020). Network anomaly detection inside consumer networks—a hybrid approach. *Electronics (Switzerland)*, 9(6), 1-12.
6. **Reddy, N. S. V., Arun, K., Mallurwar, M., & Rao, P. V.** (2023). Network Anomaly Detection Using Machine Learning: A Comprehensive Study. *International Advanced Research Journal in Science, Engineering and Technology (IARJSET)*, 10(6), 429-436.
7. **Swift.org.** (2025). *The Swift Programming Language (TSPL).* Swift Open Source Project. [Swift.org - Documentation. Diakses **20 November 2025**].