

Secure By Design

Sistem yang aman lahir dari perancangan yang matang



Anhar Solehudin
Depok, Juli 2025





About Me



Anhar Solehudin

7+ Software Engineer

Tech Lead at Digital Service Digitization
Telkom Indonesia

Ex-Tech Lead CyberArmy Indonesia

Writing Secure Coding at harscode.dev

Objective



Memiliki mindset keamanan sejak fase desain

Mengenali ancaman dengan pendekatan threat modeling

Menerjemahkan ancaman menjadi checklist dan guideline secara kolaboratif

Melatih kolaborasi lintas peran melalui studi kasus yang nyata



The Mindset

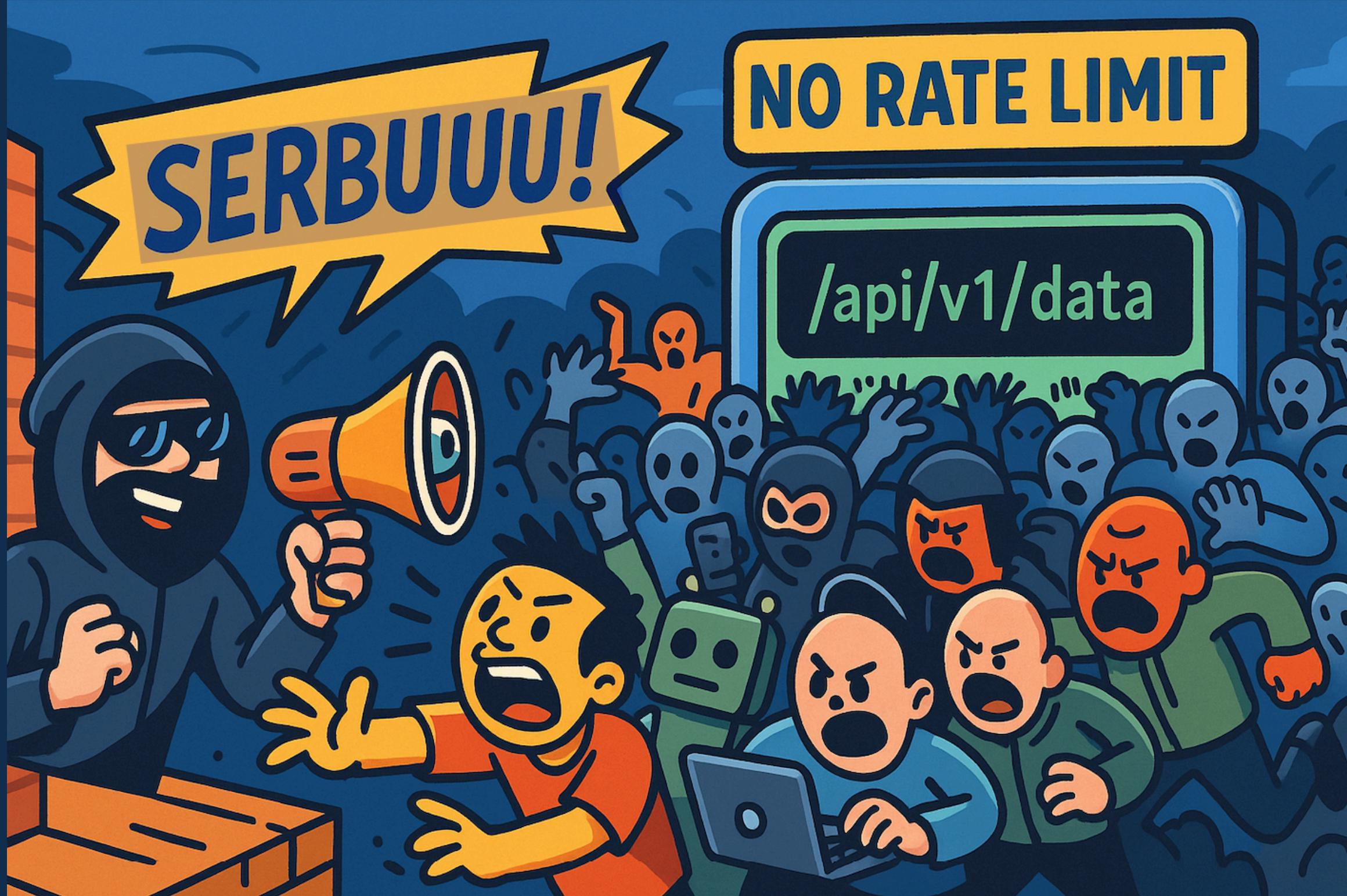


Apa itu “Secure By Design”

“Pendekatan pengembangan sistem di mana prinsip keamanan menjadi bagian integral
sejak tahap perancangan awal,
bukan sebagai pelengkap di akhir”

BAYANGKAN JIKA INI TERJADI





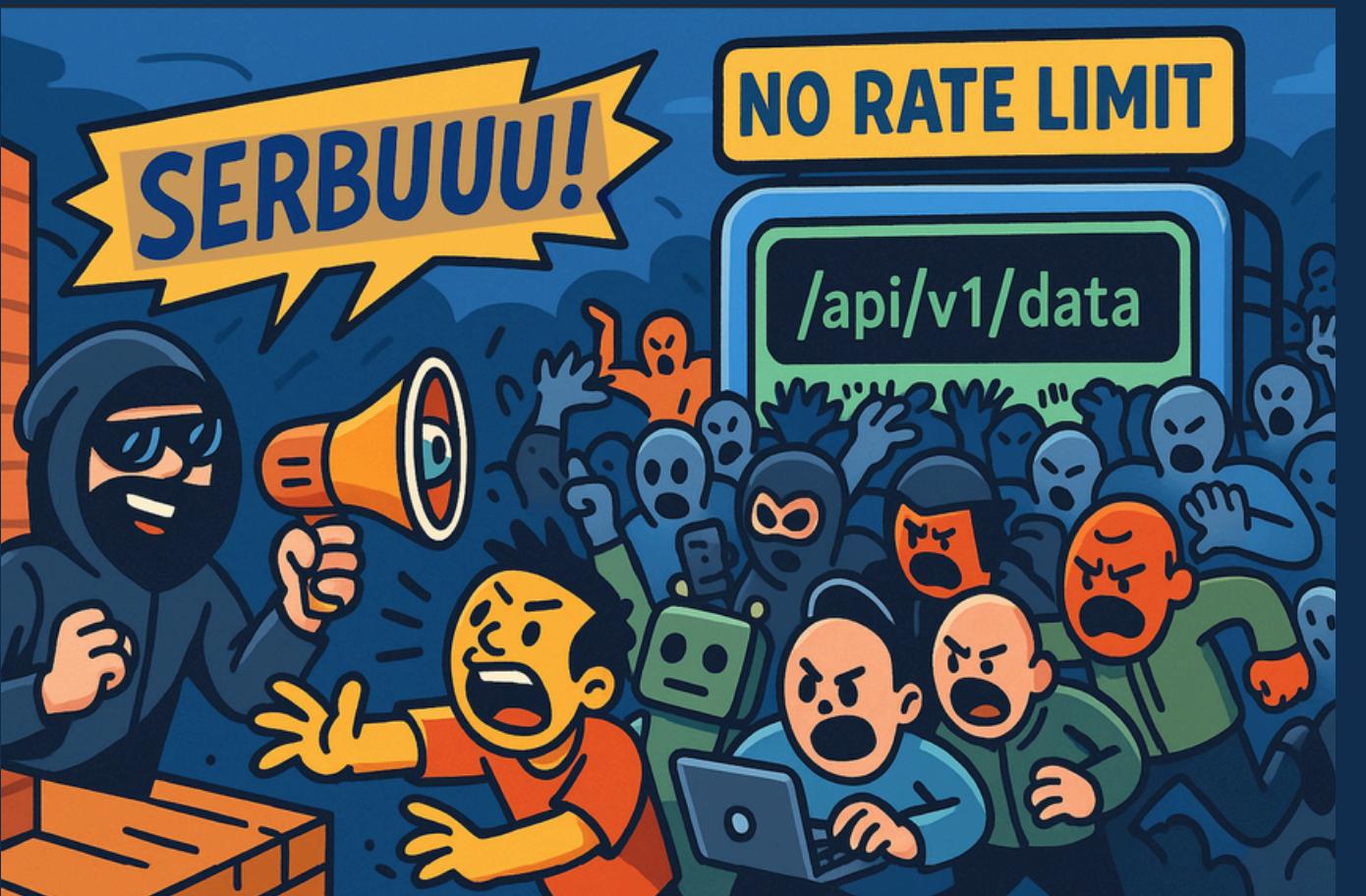
API tanpa Rate Limit
kaya Mobil Tanpa Rem

Padahal semudah ini
pasang rate limit

VS

```
use Illuminate\Support\Facades\Route;

Route::middleware('throttle:10,1')->group(function () {
    Route::post('/login', [AuthController::class, 'login']);
    Route::get('/data', [DataController::class, 'index']);
});
```





“Secure Coding itu, lebih sulit
memulai daripada mengamankan”
--- harscode



Mulai Dari mana ?



Mulai Dari Sini

- Kenapa Ada Security ?
- Kenapa Ada Polisi ?
- Kenapa Pasang CCTV ?
- Apa yang Berharga ?
- Siapa yang Rugi ?
- Memang Ada Apa ?

ANCAMAN





Threat



"A potential cause of an unwanted incident, which may result in harm to a system or organization."

---- ISO/IEC 27000

Ancaman (threat) adalah potensi aksi, celah, atau kondisi yang bisa dimanfaatkan oleh pihak tertentu untuk mencuri data, mengganggu layanan, atau merusak sistem. Ancaman bisa datang dari luar maupun dari dalam sistem itu sendiri.

Kesempatan dalam Kesempitan



Jika kamu punya kesempatan hadir di acara Bansos Kota Sejahtera,
Pikirkan apa yang akan kamu ambil, kenapa, dan bagaimana caranya.



Apa yang bisa diambil?

Uang tunai, sembako, data warga, kuota penerima, peran petugas

Kenapa itu menarik?

BerNilai langsung (uang/beras), bisa dijual, bisa digunakan untuk keuntungan pribadi

Siapa yang bisa melakukan ini?

Warga nakal, petugas internal, pihak luar yang menyamar

Dari mana bisa dilakukan?

Titik pendaftaran, antrean distribusi, jalur logistik, backdoor

Bagaimana caranya?

Pakai data palsu, daftar ganda, menyamar, suap petugas, manipulasi catatan manual

Adversarial Thinking

Security engineering requires the ability to think like an attacker to anticipate how adversaries may attempt to exploit vulnerabilities or disrupt mission objectives, and to build systems that are resilient to such attacks.

---- NIST SP 800-160 Vol.1

Adversarial thinking adalah kemampuan untuk memahami **bagaimana seorang penyerang berpikir**, mengidentifikasi potensi celah sistem, dan menggunakan pemahaman itu untuk memperkuat desain serta pertahanan sistem



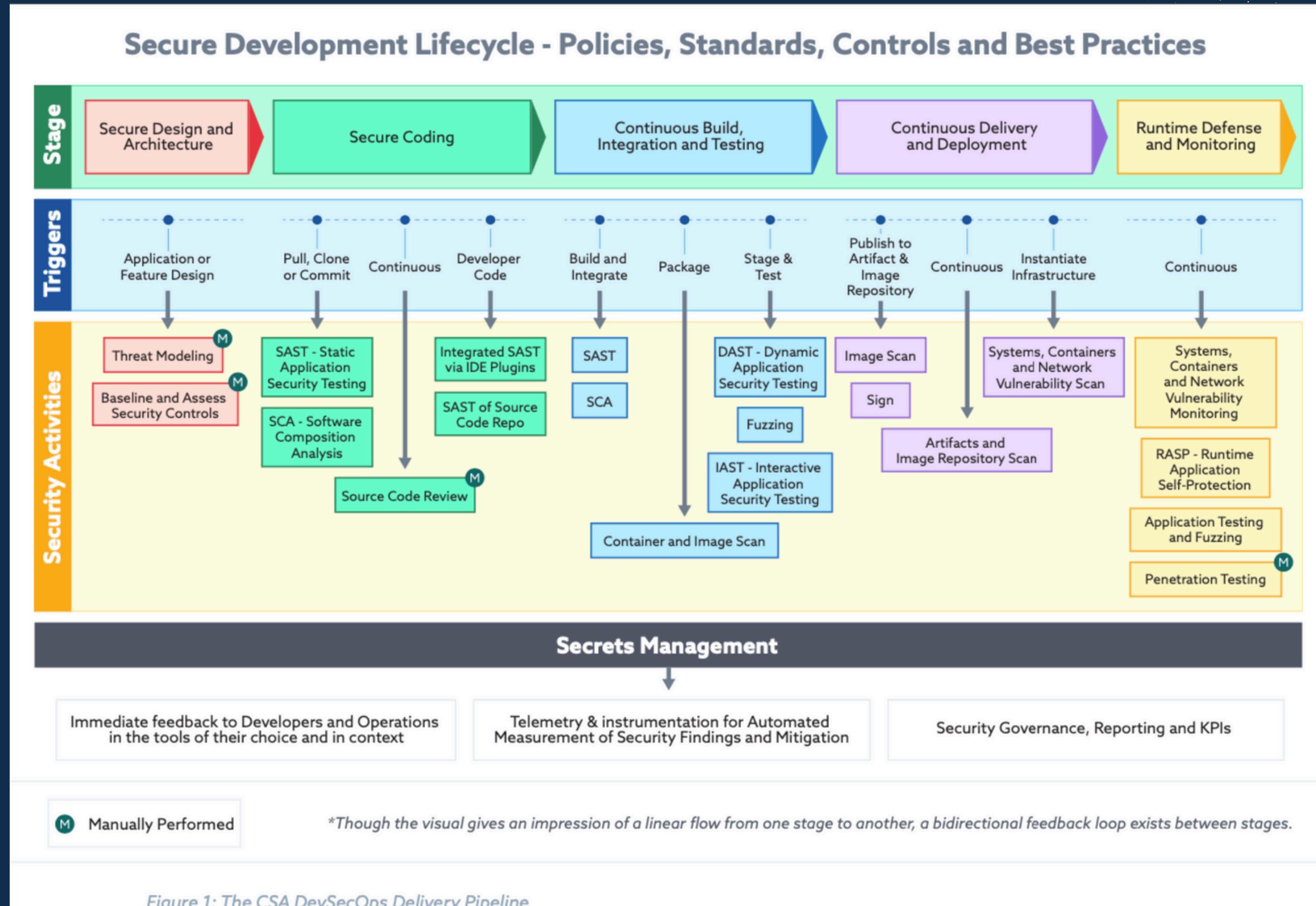


Curhat & Diskusi



Terus Gimana ?

Secure SDLC



Threat Modeling

Threat modeling is used to “**identify** and **evaluate** potential threats, vulnerabilities, and **controls** throughout the system development life cycle
--- NIST SP 800-154

Sebelum sistem dibangun, kita harus tahu:

- APA yang bisa diserang?
- KENAPA menarik untuk diserang?
- SIAPA yang bisa menyerang?
- DI MANA & DARI MANA bisa diserang?
- BAGAIMANA cara menyerangnya?



S.T.R.I.D.E.

| Category | Description |
|------------------------|---|
| Spoofing | Involves illegally accessing and then using another user's authentication information, such as username and password |
| Tampering | Involves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet |
| Repudiation | Associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations. Non-Repudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package |
| Information Disclosure | Involves the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers |
| Denial of Service | Denial of service (DoS) attacks deny service to valid users—for example, by making a Web server temporarily unavailable or unusable. You must protect against certain types of DoS threats simply to improve system availability and reliability |
| Elevation of Privilege | An unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and become part of the trusted system itself, a dangerous situation indeed |

Siapa Kamu?

Ada yang NYAMAR
jadi orang lain

Spoofing

Bukan Saya!

GA ADA BUKTI
siapa yang melakukan

Repudiation

Authentication

Integrity

Non-Repudiation

Confidentiality

Availability

Authorization

Tampering

Data Berubah!

Ada yang UTAK-ATIK
data tanpa izin

**Information
Disclosure**

Data Bocor!

Informasi sensitif
BISA DILIHAT
TANPA IZIN

**Kok Bisa
Akses?!**

GA PUNYA HAK
tapi BISA AKSES

**Denial of
Service**

**Elevation of
Privilege**

**Ga Bisa Akses
Layanan!**

Apapun cara
dilakukan supaya
LAYANAN GA BISA DIAKSES

Spoofing

Siapa Kamu?

Orang ngaku sebagai Hansip padahal bukan



Reputation

Bukan Saya!

Petugas bilang "bukan saya yang ubah penerima bansos!"



Tampering

Data Berubah!

Daftar penerima bansos diubah diam-diam



Information Disclosure

Data Bocor!

Data Pribadi Penerima Bansos Di Share di Grup WA



Ga Bisa Akses Layanan!

Antrian Palsu dari Orang Bayaran Ngabisin Jatah Bansos

Denial of Service

Elevation of Privilege

Kok Bisa Akses?!

Warga Kota Lain, Tak Terdaftar, Tapi Bisa Dapat Bansos



Implementasinya Dong!



User Story

"A user story is a brief, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system."

— Agile Alliance Glossary

User Story adalah deskripsi singkat kebutuhan atau fitur, dilihat dari perspektif pengguna, untuk menjelaskan siapa yang butuh, apa yang dibutuhkan, dan mengapa.

User Story



“Sebagai karyawan, saya ingin mengajukan reimburse agar bisa mendapat penggantian biaya operasional.”



From User Story to Security Checklist



SPOOFING

THINK

"Bagaimana kalau saya bisa ajukan reimburse pakai nama orang lain, tapi dananya masuk ke rekening saya?"

THREAT

Seorang karyawan bisa menyamar sebagai karyawan lain yang mungkin memang valid untuk reimbursement, tapi memasukkan nomor rekening pribadi agar dana cair ke dirinya sendiri

CHECKLIST

- Fitur hanya bisa digunakan setelah login yang tervalidasi
- ID karyawan tidak boleh dikirim dari frontend, harus diambil dari session
- Nomor rekening harus diverifikasi milik user yang sedang login (HRIS sync atau whitelisting)
- Audit log siapa yang submit, kapan, dan IP address/device-nya
- Jika perlu: notifikasi ke karyawan asli jika ada klaim atas namanya



TAMPERING

THINK

"Bagaimana kalau saya bisa ubah nominal klaim atau nomor rekening sebelum data itu sampai ke sistem?"

THREAT

Seorang karyawan memanipulasi data reimburse dari sisi frontend atau di perjalanan (intercept request), misalnya:

- Mengubah nominal dari 500.000 menjadi 5.000.000
- Mengganti nomor rekening agar dana masuk ke akun lain
- Mengubah jenis pengeluaran agar otomatis disetujui (misal dari "lain-lain" ke "transport resmi")

CHECKLIST

- Semua input penting (nominal, rekening, jenis pengeluaran) harus divalidasi ulang di backend
- Gunakan data master untuk field seperti jenis pengeluaran (bukan teks bebas)
- Cegah manipulasi request dengan validasi signature/token
- Batasi nilai maksimum klaim dan jenis transaksi yang diizinkan per role
- Implementasi integrity check untuk mendekripsi perubahan payload



REPUDIATION

THINK

"Bagaimana kalau saya bisa bilang 'Itu bukan saya yang submit klaim ini' dan sistem gak bisa membuktikan sebaliknya?"

THREAT

Seorang karyawan mengajukan klaim reimburse (bahkan klaim palsu), lalu mengelak saat diproses atau ketahuan.

Karena:

- Sistem tidak mencatat siapa yang login saat submit
- Tidak ada timestamp, IP, atau device log
- File bukti bisa diganti setelah di-upload tanpa jejak versi sebelumnya
- Approval dilakukan tanpa jejak siapa yang menyetujui

CHECKLIST

- Simpan audit log lengkap setiap pengajuan (user ID, waktu, IP/device)
- Gunakan immutable log atau append-only storage untuk klaim & approval
- Catat siapa yang melakukan approval dan kapan
- Tampilkan riwayat perubahan file atau klaim (versioning / hash)
- Kirim notifikasi ke user setiap kali klaim diajukan atas namanya



INFORMATION DISCLOSURE

THINK

"Bagaimana kalau saya bisa melihat data klaim dan nomor rekening orang lain?"

THREAT

Seorang karyawan atau pihak luar dapat mengakses data klaim yang bukan miliknya, seperti:

- Melihat nominal klaim dan bukti struk milik rekan kerja
- Mengintip nomor rekening penerima yang bersifat sensitif
- Mengakses file upload (struk/faktur) langsung lewat URL tanpa autentikasi
- Menerima email atau notifikasi klaim yang dikirim ke alamat yang salah
- API menampilkan field internal yang tidak seharusnya ada di response (contoh: ID rekening atau hash internal)

CHECKLIST

- Terapkan Access Control (hanya pemilik dan approver yang bisa lihat klaim)
- Lindungi file upload (struk/faktur) dengan signed URL atau private storage
- Masking atau enkripsi untuk field sensitif seperti nomor rekening
- Validasi email tujuan sebelum mengirim notifikasi
- Minimalisasi data di response API (hanya field yang diperlukan)



DENIAL OF SERVICE

THINK

"Bagaimana kalau saya spam form reimburse ratusan kali atau upload file super besar sampai sistem lumpuh?"

THREAT

Seorang karyawan atau pihak luar mencoba mengganggu ketersediaan sistem, misalnya:

- Mengirim ratusan klaim dalam waktu singkat (flood request)
- Upload file bukti sangat besar berulang-ulang sampai storage penuh
- Membuat script bot untuk submit klaim otomatis terus menerus
- Melakukan request status klaim dalam loop cepat hingga server overload
- Menyalahgunakan API open endpoint untuk membanjiri notifikasi approval

CHECKLIST

- Terapkan rate limiting untuk endpoint submit dan status klaim
- Batasi ukuran file upload & validasi tipe file
- Gunakan CAPTCHA atau mekanisme anti-bot di form publik
- Monitor & alert untuk traffic abnormal (spike detection)
- Isolasi proses intensif agar tidak mengganggu fungsi utama (misalnya pakai queue)



ELEVATION OF PRIVILEGE

THINK

"Bagaimana kalau saya yang bukan atasan bisa ikut approve klaim atau bahkan mengubah klaim orang lain?"

THREAT

Seorang karyawan biasa memperoleh akses atau hak yang seharusnya tidak dimiliki:

- Bisa mengakses fitur approval meskipun bukan supervisor/HR
- Mengubah status klaim karyawan lain melalui API (karena backend tidak validasi role)
- Menemukan endpoint tersembunyi (hidden admin route) yang tidak dicek perannya
- Mengubah role di token/session (misalnya mengedit JWT menjadi "role":"admin")
- Menggunakan akun orang lain yang punya hak lebih karena session tidak dilogout

CHECKLIST

- Validasi role & hak akses di backend, bukan hanya sembunyikan tombol di frontend
- Gunakan token/session yang ditandatangani & divalidasi setiap request
- Audit akses setiap kali ada perubahan hak/role atau approval dilakukan
- Pisahkan endpoint admin/supervisor dengan akses normal
- Terapkan session timeout & proteksi reuse token



**Udah merasa terancam nih!
Sekarang harus ngapain ?!**

Threat Models



User Story



Security Checklist



Secure Coding
Guidelines

Secure Coding Guidelines

1. Security Checklist
2. Prinsip & Standar
3. Acceptance Criteria
4. Contoh Implementasi Teknis





Secure Coding Guidelines : Security Checklist

Checklist :

“Terapkan rate limiting untuk endpoint submit dan status klaim.”

Detail :

Batasi jumlah request ke endpoint klaim (/submit-claim, /status-claim) agar tidak bisa disalahgunakan untuk:

- Spam klaim (DoS)
- Brute force pengecekan status klaim
- Penyalahgunaan resource sistem



Secure Coding Guidelines : Prinsip & Standar

Prinsip :

“Prevent Denial of Service (DoS) & Unrestricted Resource Consumption”

Referensi :

- OWASP API Security Top 10 2023 [API4: Unrestricted Resource Consumption]
- OWASP Cheat Sheet – Rate Limiting
- OWASP ASVS 5.3.5: Verify APIs implement rate limiting to prevent abuse

Secure Coding Guidelines : Acceptance Criteria

Criteria :

1. **Limit Request Per IP** : Mengirim >5 request ke endpoint /submit-claim dalam 1 menit → response 429 Too Many Requests.
2. **UI Protection** : Tombol submit otomatis disable selama proses submit berlangsung.
3. **Behavior Normal** : Submit 1–3 kali dalam kondisi normal tetap sukses (200 OK).
4. **Logging & Monitoring** : Event throttle dicatat di log dan memicu alert jika terjadi berulang.

Secure Coding Guidelines : Contoh Implementasi Teknis Backend

php

 Copy  Edit

```
// routes/api.php
use App\Http\Controllers\ClaimController;

Route::middleware('throttle:5,1')->group(function () {
    Route::post('/submit-claim', [ClaimController::class, 'submit']);
    Route::get('/status-claim', [ClaimController::class, 'status']);
});
```

Tambahkan middleware throttle untuk membatasi request ke endpoint klaim:

- throttle:5,1 → Maksimum 5 request per menit per IP.
- Bisa dikonfigurasi lebih ketat sesuai kebutuhan (misal 3,1 untuk submit klaim).

Secure Coding Guidelines : Contoh Implementasi Teknis Frontend

Tambahkan proteksi klik berulang pada tombol `Submit`:

```
<template>
  <button :disabled="isSubmitting" @click="submitClaim">
    Ajukan Klaim
  </button>
</template>
```

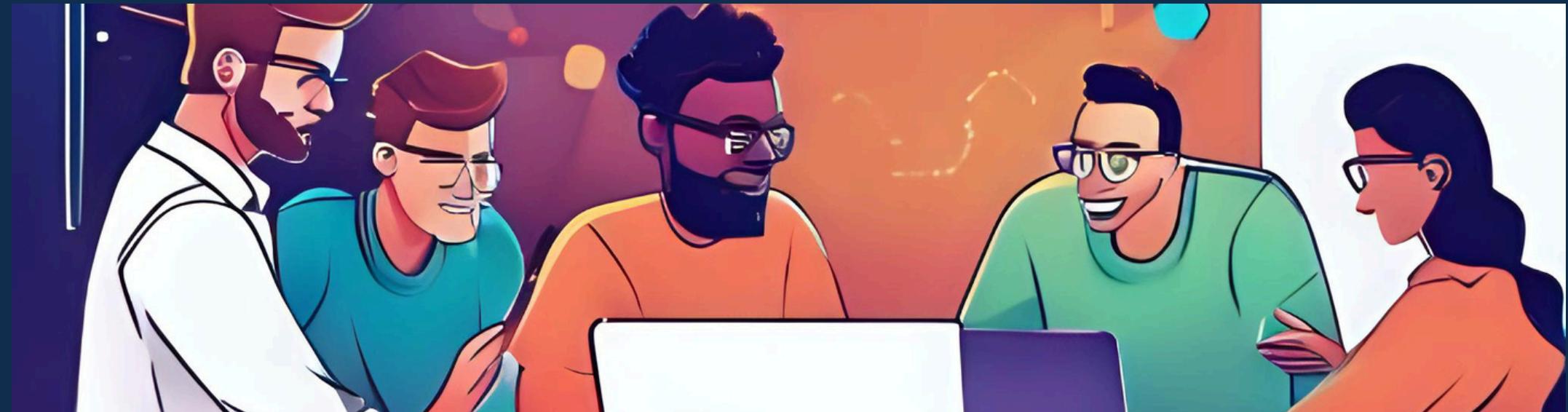
```
<script>
export default {
  data() {
    return { isSubmitting: false };
  },
  methods: {
    async submitClaim() {
      this.isSubmitting = true;
      try {
        await this.$axios.post('/api/submit-claim', this.formData);
        this.$toast.success("Klaim berhasil diajukan");
      } catch (error) {
        this.$toast.error(error.response.data.message || "Gagal submit");
      } finally {
        // Re-enable tombol setelah 1 detik
        setTimeout(() => { this.isSubmitting = false; }, 1000);
      }
    }
};
</script>
```





Curhat & Diskusi

Summary

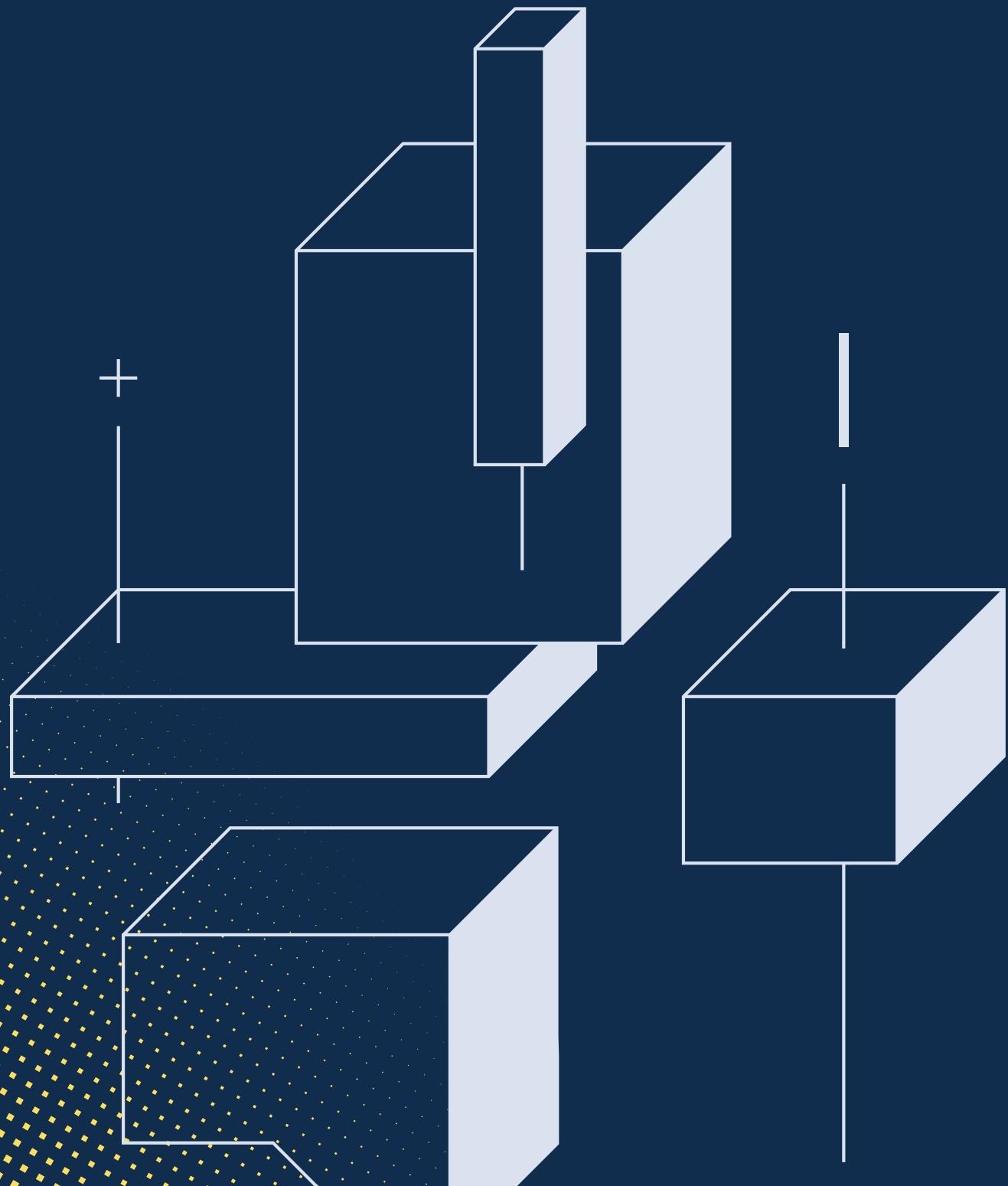


Pahami dulu apa yang akan dibangun, dan kenapa “berharga”

Pikirkan Ancaman-Ancaman sejak awal yaitu pada fase design

Gunakan adversarial thinking & threat modeling framework seperti STRIDE.

Jadikan Standar, Dokumentasikan, Terapkan, Evaluasi, Iterasi



Terima Kasih



www.harscode.dev