CYBERSECURITY
&
SECURE PROGRAMMING

Threat Modelling

The question that wasn't asked

Optus, September 2022

- ~9.8 million current and former Optus customers exposed.
- Names, dates of birth, addresses, passport and driving-licence numbers.
- The technique took no skill, no malware, no zero-day.
  - An internet-facing API at /users/{userId}.
  - Attacker incremented the number and called it again.
  - Endpoint required no password, no token, no identity check.
- ACMA's 2024 court filing: "not highly sophisticated… a simple process of trial and error".

CYBERSECURITY
&
SECURE PROGRAMMING

DISCUSSION

What did the Optus engineers fail to ask at design time?

The Four Questions That Save You Time

Shostack's Four Question Frame

- 1. What are we working on?
- 2. What can go wrong?
- 3. What are we going to do about it?
- 4. Did we do a good enough job?
- Wording is canonical — Shostack asks it not be paraphrased.
  - The collective 'we' is intentional: a team activity, not done TO a team.
  - Question 4 is the one teams skip — and the one that matters most.

Working definition

Threat Modelling

The practice of answering Shostack's four questions, with enough rigour and enough people in the room to produce decisions you can act on. It is a conversation more than a document. The output you want is a list of changes the team makes to the design BEFORE writing the code.

— adapted from Shostack, 2024

CYBERSECURITY
&
SECURE PROGRAMMING

Drawing the System — DFDs

Four shapes and one annotation

- PROCESSES (circles) — running code: a web server, a Lambda, a stored procedure.
- DATA STORES (parallel lines) — data at rest: Postgres, S3, Redis, a cookie, a log.
- EXTERNAL ENTITIES (squares) — anything outside your control: users, third-party APIs.
- DATA FLOWS (labelled arrows) — always directional, always labelled with WHAT flows.
- TRUST BOUNDARIES (dashed lines) — where the level of trust changes.

The element students miss

Trust Boundary

A line on the diagram across which the level of trust changes. Crossing a trust boundary is where threats live: it is the place where input must be validated, identity must be authenticated, and authorisation must be checked. A DFD without trust boundaries is decorative.

DFD vs architecture diagram — the three-test rule

- Arrows are ALWAYS directional.
- Arrows are ALWAYS labelled with what flows.
- Trust boundaries are ALWAYS present.
- Fail any of the three? It's an architecture diagram, not a DFD.
- Draw at Level 1 — one decomposition below the system box. Going deeper produces threats nobody fixes.

CYBERSECURITY
&
SECURE PROGRAMMING

STRIDE — Six Ways Things Go Wrong

STRIDE — Microsoft, 1999

- S — Spoofing → violates AUTHENTICATION.
- T — Tampering → violates INTEGRITY.
- R — Repudiation → violates NON-REPUDIATION.
- I — Information disclosure → violates CONFIDENTIALITY.
- D — Denial of service → violates AVAILABILITY.
- E — Elevation of privilege → violates AUTHORISATION.

CYBERSECURITY & SECURE PROGRAMMING

Live STRIDE pass — Moodle assignment submission

Walk all six letters on Flow 1 (student → front-end). Take threats from the floor for at least two letters.

S — Spoofing on Flow 1

- Could someone submit an assignment as another student?
- Yes, if…
  - The session cookie can be stolen via XSS in a forum post.
  - The cookie isn't bound to the user's device.
  - The front-end accepts a user_id from the request body.
- Mitigation.
  - HTTP-only, Secure, SameSite cookies.
  - Never trust a user identifier from the request body for an authenticated action.
  - Re-auth for high-stakes actions like final submission.

T — Tampering on Flow 1

- Could the student modify the file en route, or alter request fields the server should compute?
- Yes, if…
  - TLS is missing or terminated too early.
  - The handler trusts a 'grade' or 'status' field from the request.
- Mitigation.
  - TLS 1.3 end-to-end.
  - Never accept fields the server should compute itself.
  - Validate on the server, not the client.

R — Repudiation on Flow 1

- Six weeks later: 'I submitted on time, your system lost it.' Can you prove what happened?
- Only if you logged…
  - Server timestamp.
  - Originating IP.
  - SHA-256 of the file.
  - An immutable, append-only audit trail.
- Without that, you're arguing about the time on someone's laptop.

I — Information Disclosure on Flow 1

- Could one student read another's submission?
- Yes, if /submission?id=12345 isn't owner-checked.
  - This is the Insecure Direct Object Reference (IDOR) pattern.
  - Single most common authorisation bug in web applications (OWASP A01).
  - It is the bug at the heart of the Optus 2022 breach.
- Mitigation.
  - On every read, check the requester owns the resource (or has lecturer privileges).
  - Enumerate what the SERVER should compute, not what the CLIENT should send.
- We treat IDOR/BOLA in depth in Chapter 17.

D — Denial of Service on Flow 1

- Could a student submit a 50-GB file the night before the deadline?
- Could they submit a 'zip bomb' that explodes from 1 KB to 100 GB during AV scanning?
- Mitigation.
  - Maximum upload size enforced at the load balancer, before app code runs.
  - Cap the AV scanner's expansion budget.
  - Rate-limit per-user.

E — Elevation of Privilege on Flow 1 (and 5)

- Could a crafted upload give the student arbitrary code execution on the server?
- The Equifax-class threat (CVE-2017-5638, Struts multipart parser).
  - Vulnerable parser sitting on a server-side trust boundary.
- Could a path-traversal in the filename let them write outside /submissions/? (Flow 5.)
- Mitigation.
  - Normalise and validate filenames server-side.
  - Never use a user-supplied filename as a path component.
  - Patch parsers; least-privilege the file-system permissions of the process.

Optus 2022 — the cleanest single-letter case study

- S (Spoofing) — 'who can call this endpoint, and how do we know they are who they say they are?' — never asked.
- I (Information Disclosure / IDOR) — 'if a user can request resource N, what stops them requesting N+1?' — never asked.
- Either question, asked once during a 30-minute review, would have caught the bug.
- An authentication check existed; a 2018 refactor silently broke it. That's the case-in-chief for Question 4.

CYBERSECURITY
&
SECURE PROGRAMMING

Other Lenses You Might Need

Which lens, when?

Lens	When to reach for it
STRIDE	Default. Any system with code, users and data flows.
LINDDUN	Add when the system handles personal data (almost always).
Attack Trees	Add when 'could the attacker do X?' is the question.
PASTA	Use when business risk drives the conversation (regulated industries).
MITRE ATT&CK	Threat library — feeds Q2, not a methodology in itself.
DREAD	Cautionary tale. Microsoft retired it ~2008. Don't use.

CYBERSECURITY
&
SECURE PROGRAMMING

Where Threat Modelling Fails

Heartbleed (CVE-2014-0160) — the design/implementation gap

- RFC 6520 specifies that an oversized HeartbeatMessage MUST be discarded silently.
- The protocol was threat-modelled correctly. The bug was a missing length check in C.
- At disclosure (7 April 2014), ~17% of TLS-protected web servers were exposed.
- TM addresses the design layer; it does not replace memory-safety, fuzzing or static analysis.
- After Heartbleed, OpenSSL was added to OSS-Fuzz. We pick this up in Chapter 16.

Capital One 2019 — three components that looked fine in isolation

- March/April 2019: SSRF in a misconfigured ModSecurity WAF on EC2.
- The chain.
  - WAF threat model: 'filter HTTP'.
  - EC2 role threat model: 'I'm a service account for the WAF'.
  - IMDSv1 threat model: 'anything on this VM is trusted to ask'.
- Result: temporary IAM credentials with s3:GetObject on 700+ buckets — including 106M credit-card application records.
- STRIDE Elevation of Privilege on the WAF → metadata-service flow is exactly the prompt that surfaces it.

Five mistakes to recognise before you make them

- Drew a network diagram and called it a DFD.
- Skipped the trust boundaries.
- STRIDE'd everything to death — 200 unprioritised threats, no owners.
- No developers in the room — security wrote it, devs ignored it.
- Done once, never updated. By Q3, it's fiction.

Recap

1. Most public breaches would have been caught by one structured question, asked once.
2. Shostack's Four Questions are the spine; STRIDE/LINDDUN/attack trees/PASTA all answer Q2.
3. DFDs need trust boundaries. Without them the diagram is decorative.
4. STRIDE = six security properties the system is supposed to have. Walk every flow.
5. TM fails when assumptions silently shift, when design TM meets implementation bugs, or when it becomes paperwork.

Try This — bring to the W2 practical

- Pick a web app you use daily — your bank, your email, your university portal.
- Sketch its login form as a one-flow DFD.
- Run STRIDE on the single flow from user to login service.
- Aim for one concrete threat per letter, ten minutes total.
- You will surprise yourself with how many threats you've already heard of in news stories.

What's next

- W2 L2: Frameworks Worth Knowing — NIST CSF 2.0, ISO 27001, GDPR, NIS2 in Ireland.
- W2 Practical: threat-model OWASP Juice Shop with Threat Dragon, run STRIDE on its DFD, raise findings as tickets.
- Reading: Chapter 4 of the book; the Threat Modeling Manifesto (~800 words) if you want one external link.
- Bring your Try This sketch to the practical — we'll compare them.