Threat Modelling on OWASP Juice Shop

Framing — minutes 0 to 10

Shostack's Four Questions — back on the board

- Q1. What are we working on?
- Q2. What can go wrong?
- Q3. What are we going to do about it?
- Q4. Did we do a good enough job?
- Today's flow: Q1 = scope + DFD; Q2 = STRIDE; Q3 = decisions; Q4 = retrospective in the write-up.

Pin this scope statement to the top of your paper

```
System under study : OWASP Juice Shop v19.2.1
Slice in scope     : Auth + basket + checkout
                     (registration to order placement)
Out of scope       : Complaint forms, chatbot, B2B XML,
                     delivery tracking, the Score Board
Assumptions        : Single instance, default config,
                     no WAF, vanilla install
```

Four lines. Three named assumptions. Copy mine; do not improvise.

CYBERSECURITY
&
SECURE PROGRAMMING

Setup — start the lab

CYBERSECURITY & SECURE PROGRAMMING

Start Juice Shop locally — bind to localhost

```
$ docker run --rm -p 127.0.0.1:3000:3000 bkimminich/juice-shop
...
info: Server listening on port 3000

# Verify in a second terminal:
$ nmap -sV -p 3000 127.0.0.1
$ curl -s http://localhost:3000/api/Challenges \
      | jq '.data | length'
111
```

-p 127.0.0.1:3000:3000 — not -p 3000:3000. Localhost only.

The diagramming tool — pick one before we draw

- OWASP Threat Dragon v2.6.0 — purpose-built; STRIDE, LINDDUN, CIA, DIE, PLOT4ai engines built in.
- Web app at threatdragon.com, Electron desktop binary, or lab Docker fallback.
- Backup options if Threat Dragon refuses to install:
  - draw.io — browser, no login.
  - Excalidraw — browser, real-time collaboration.
- Pencil and paper for the FIRST twenty minutes. Non-negotiable.
- Switch to a tool only when the diagram is stable.

CYBERSECURITY
&
SECURE PROGRAMMING

EoP card game — minutes 10 to 25

Elevation of Privilege — rules for a 15-minute hand

- 78 cards, six suits, one suit per STRIDE letter. Tampering is trumps.
- 2 to 6 players per table; deal evenly.
- Player holding the 3 of Tampering leads the first trick.
- On your turn, play one card, following suit if you can.
- To score: point at a DFD element and explain how the threat on the card applies to Juice Shop.
- If the table agrees, you win the trick and one point. Scribe writes it down.

CYBERSECURITY
&
SECURE PROGRAMMING

Draw the DFD — minutes 25 to 55

The non-negotiable: trust boundaries

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. Juice Shop has four meaningful boundaries; your DFD must show at least the first three.

— Chapter 4, recapped

Juice Shop DFD — what goes on the page

- External entities (squares):
  - E1 anonymous visitor; E2 registered customer; E3 admin; E4 mock payment endpoint.
- Processes (circles):
  - P1 Angular SPA; P2 Auth API; P3 Basket API; P4 Coupon engine; P5 Order/Payment.
- Data stores (parallel-line boxes):
  - D1 SQLite; D2 ftp/ receipts directory; D3 jwt.key — RS256 private key in the source tree.
- Trust boundaries (dashed):
  - TB1 Internet/server; TB2 server/DB; TB3 server/payment; TB4 server/filesystem.
- Fifteen labelled, directional flows tie it all together.

Three rules for an arrow on the page

- Every arrow is directional — arrowheads, not lines.
- Every arrow is labelled with what flows — JSON, SQL, JWT, PDF.
- Every arrow either crosses a trust boundary, or it isn't interesting for STRIDE.
- If your diagram could appear unchanged in a marketing deck, it isn't a DFD.
- Repeat at minute 25 and again at minute 35. Pairs who land this rule produce twice the threats.

STRIDE pass — minutes 55 to 95

The deliverable per pair, this phase

- Two to four threats, each:
- (1) names the DFD element or flow it sits on,
- (2) states the threat in one sentence — 'could someone …?',
- (3) picks the STRIDE letter (and a second one if it genuinely violates two),
- (4) maps to a SPECIFIC Juice Shop challenge the pair can reproduce.
- Step 4 is the pedagogical innovation — it converts threat modelling from paper to verified.

Seven headline threats from the answer key

ID	Letter	Threat	Challenge
T1	S	SQLi in login: ' OR 1=1-- returns admin row	loginAdminChallenge
T7	T	Cross-user basket POST via client BasketId	basketManipulateChallenge
T11	R/E	JWT signing key checked into the source tree	jwtForgedChallenge
T13	I	View-basket IDOR — increment the URL ID	basketAccessChallenge
T18	D	NoSQL $where infinite loop hangs the DB thread	noSqlCommandChallenge
T21	E	Mass-assigned 'role' field on registration	registerAdminChallenge
T22	E	/#/administration guarded only by client JS	adminSectionChallenge

The marquee — T11, JWT key in source

The RSA private key for JWT signing is in the public repository.

encryptionkeys/jwt.key — visible to any attacker with internet access.

Find the leaked key — and forge a token with it

```
# 1. Clone the upstream repo (no exploit yet — just reading).
$ git clone https://github.com/juice-shop/juice-shop.git
$ cd juice-shop
$ find . -name '*.key' -o -name 'jwt.*'
./encryptionkeys/jwt.key
./encryptionkeys/jwt.pub

# 2. Inspect the running JWT (decode the middle segment).
$ echo 'eyJhbGciOi...' | cut -d. -f2 | base64 -d | jq

# 3. Edit the email claim, re-sign with the leaked key,
#    present the new token. Score Board lights up.
```

The 'oh' moment: the key on disk in your container is the same key on github.com.

The fix is elimination, not mitigation

Eliminate vs Mitigate

T11 is not mitigated by adding rate limits, MFA or a WAF. It is eliminated by removing the key from version control entirely, rotating it, putting it behind a KMS (AWS KMS, HashiCorp Vault, GCP KMS) per environment, and scrubbing the git history. The four-way decision matters here: 'Eliminate' is the right verb.

— Chapter 6, section 6.3

Report-out and decisions — minutes 95 to 110

Three rules for the decision column

- No 'TBD'. A row without a decision is a row where the threat model has failed.
- Force the four-way choice on every threat:
  - Mitigate — add a control.
  - Eliminate — remove the design fault.
  - Transfer — push the risk to a third party (insurance, processor).
  - Accept — document and move on, with a sunset date.
- Every row needs an owner — a role, not 'the team'. 'Backend lead', 'ops', 'product'.

A working consolidated table — what 'good' looks like

ID	Sev	Owner	Decision
T1	H	Backend	Mitigate — parameterised query
T7	H	Backend	Mitigate — derive BasketId from JWT
T11	C	Sec/Ops	Eliminate — rotate, KMS, scrub history
T13	H	Backend	Mitigate — server uses JWT subject
T18	M	Backend	Mitigate — disable $where, query timeout
T21	H	Backend	Mitigate — force role server-side
T22	H	Backend	Mitigate — server-side authz on every admin endpoint

CYBERSECURITY
&
SECURE PROGRAMMING

Live exploit — minutes 110 to 120

CYBERSECURITY & SECURE PROGRAMMING

Three exploits, ten minutes, Score Board lights up

```
# T1 — Login as admin, no password.
Email:    ' OR 1=1--
Password: anything

# T13 — View someone else's basket (IDOR).
GET /rest/basket/6   # logged in as basket-id 7
# decrement the path id; previous user's basket comes back.

# T11 — Forge a JWT with the leaked key.
$ jwt encode --secret-file encryptionkeys/jwt.key \
      --alg RS256 \
      '{"email":"jwtn3d@juice-sh.op","role":"admin"}'
```

The point isn't to cheat the lab. It is to make the loop visible — identify, decide, verify.

CYBERSECURITY
&
SECURE PROGRAMMING

The deliverable

What goes in the PDF

- Cover — pair names, student numbers, scope statement, assumptions. (~half page.)
- DFD — full-page Level-1, exported from Threat Dragon / draw.io / Excalidraw, four trust boundaries. (1 page.)
- Threat list — table of 15 to 20 rows, every row carrying a STRIDE letter and a Juice Shop challenge key. (1-2 pages.)
- Mitigation notes — top five threats by severity, one paragraph each, traceable to OWASP ASVS or a CWE. (1 to 1.5 pages.)
- Reflection — one page on the three retrospective prompts.
- Filename: comp09031-w2-<surname1>-<surname2>.pdf. Submit through Moodle.

The rubric — how the tutor will mark this

Component	Weight	What 'good' looks like
Scoping	10%	Clear in/out scope. Three named assumptions.
DFD quality	20%	All four shapes. Directional, labelled arrows. >=3 trust boundaries.
STRIDE coverage	40%	Threat per letter on >=3 elements. Every row carries a challenge key.
Mitigation realism	20%	Specific control named (header, cookie flag, IAM policy). Not 'validate input'.
Reflection	10%	One page: surprises, what you'd do differently, an unwritten assumption.

Five common mistakes — recognise them before you make them

- Boiling the ocean. Modelling all 111 challenges instead of the in-scope slice.
- Architecture diagram, not DFD. Lines without arrowheads, flows without labels, no boundaries.
- Skipping trust boundaries. Every flow looks the same; high-leverage threats vanish.
- Threat-list-without-decisions. Eighteen threats, no owners, no challenges. A wishlist.
- No retrospective. Question 4 gets skipped; the model never improves.

Takeaways from the practical

1. Identify the threat, then prove it — every row maps to a Juice Shop challenge you can reproduce.
2. A DFD without trust boundaries is decorative. Cross a boundary; that is where threats live.
3. STRIDE is the engine, not the goal. The goal is the four-way decision: Mitigate, Eliminate, Transfer, Accept.
4. Secrets in source is elimination, not mitigation. T11 is the textbook example.
5. Question 4 is what threat modelling fundamentally is for. Don't skip the retrospective.

What's next

- Tonight: write the 4-6 page PDF while the session is fresh.
- Friday 17:00: submit through Moodle as comp09031-w2-<surname1>-<surname2>.pdf.
- Next lecture (W3 L1): Cryptography foundations — the maths and primitives behind T11.
- Optional: Pwning OWASP Juice Shop guide — use AFTER the session, not during.
- Questions on the website forum or in office hours.