

ShopNow

Building a better tomorrow for buyers and vendors
(especially vendors)



Identify criticality, bottlenecks	2
Context of deployment	2
Sensible data handling	2
Identifying sensible technical components	2
Data flow	3
Security zones	3
High-privilege accounts	4
Critical data bottlenecks	4
High-sensitivity assets	4
Exposition of an asset when combined	4
Prediction of failures and vulnerabilities (DRP)	4
Report conclusions	5
STRIDE analysis	6

Identify criticality, bottlenecks

Context of deployment

As an e-shopping platform, ShopNow ought to deliver the best and fastest

In the user experience, from browsing to shipping, a several amount of data of various criticality nature and processed client and server-side. To ensure the best practices and the best outcome, ShopNow has missionned

Sensible data handling

Considering the data handled within ShopNow, sensitive data includes :

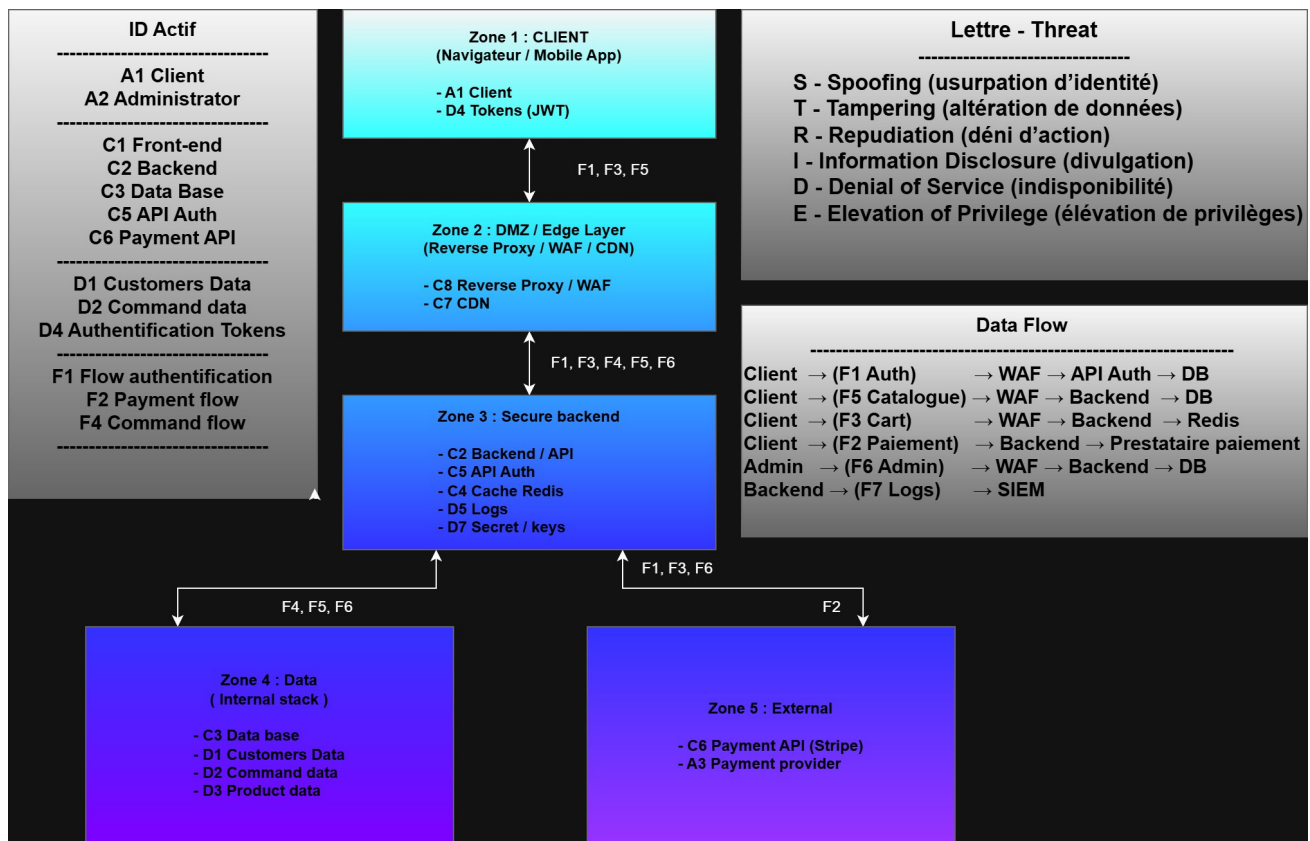
TYPE OF DATA	SENSIBILITY NATURE	ASSOCIATED RISKS
Client data	Name, Email, Physical address (GDPR type)	Phishing campaigns
Details of order	Date, Price total	Phishing campaigns
Authentication tokens	JWT tokens (for API handling) and refresh token (for aborted sessions without interrupting shopping flow)	Credentials stuffing
Payment data	Transactionnal references	
Keys and secrets	API keys, deployment secrets	CI/CD poisoning
Backend	Codebase integrity	
Databases	Customers, Logins, Products and Invoices storage	
Auth API	Login sessions and tokens	Credentiels stuffing
Payment API	STRIPE	
Payment flow	Transactions tokens and banking data	

Identifying sensible technical components

Technical components constituting infrastructre backbone includes :

SENSIBLE TECH COMPONENTS	BUSINESS LOGIC
Backend / API	The product core before frontend
Authentication API	Handling authentication (sessions, tokens, access rights)
Databases (PostgreSQL, MySQL)	Sensible data
Reverse Proxy Server	Requests filtration and redirection
WAF	Web attacks handler and alerter
Payment API	
Endpoints	
CDN	Products displaying per region

Data flow



Sensible data flow includes flows processing sensible data, and thus needing special security or secrecy.

ID	SENSIBLE DATA FLOW	FLOW PURPOSES
F1	Authentification	Everything related to login, refreshing, logout
F2	Payment	Payment third-parties connected to backend
F6	Administration	Products stock, prices, sales
F7	Logs	Logging processing and storage

Security zones

Security zones must be isolated considering the need of data integrity and external inaccessibility. Zones to isolate includes :

SECURE ZONES	WHY
Backend	<ul style="list-style-type: none"> No need to access it, frontend should only get what's needed without direct access (endpoints dedicated with secrets handling) Access to backend may lead to security leaks and industrial intelligence
Databases	Customers data, products hidden metadata and orders details should remain private and isolated for special and dedicated access only

High-privilege accounts

ID	SECURE ZONES	WHY
A2	Local administrators	Admin access on backend, databases
A4	(IN CASE OF K8S CLUSTER FOR INSTANCE) Crons & Workers	Root-access level of jobs and POF capacities

Critical data bottlenecks

In case of compromise of any asset in the pipeline, the business as a whole may be impacted. So far, here are the actors which could lead to business disruption if poisoned :

ASSET	CONSEQUENCES
Client data	Loss of reputation : business volume shrinking
Details of order	Phishing material
Authentication tokens	May cause impersonation
Payment data	Loss of reputation, Phishing material, Scaming schemes
Keys and secrets	Requests overflow, API usurpation, Denial Of Service
Databases	Customers data loss, Products data loss, Orders details loss
Auth API	May cause impersonation

High-sensitivity assets

Some assets never should have more access than absolutely needed for the production and exploitation pipelines. Zero Trust policy should include :

ASSETS TO NERF	AVOIDED CONSEQUENCES
Databases	Avoiding wide Read & Write (and defining Read-Only if no Write is needed) on unnecessary and/or unrelated resources avoid harmful manipulations, either internal or external
Internal services (cron, jobs)	According the correct rights to jobs avoids giving root-level access to entire stack in case of usurpation
API (one for each service, no monolithic API)	Each API for it's token generate conteneurization and thus better control on data flows
Contractors	No extra permissions for third-parties external to avoid unwanted and untrackable behaviors

Exposition of an asset when combined

ASSETS...	...COMBINED WITH...	...LEADS TO
Payment API	Payment data	Fraud and financial prejudices

Prediction of failures and vulnerabilities (DRP)

Report conclusions

Additions

- Load-balancer (for regionalisation access)
- MFA for clients
- MFA for internal products
- Backup plan
- Zero-Trust by design
- Resources monitoring
- Email analyser (like Mail-In-Black for internal)
- Cold storage solution

Changes

- (if API are external)
- (if API are internal)
- Token expiration
- Internal Passwords expiration
- Hashing data in all flows, including non-sensible ones

Deletions

- DMZ (no real use)

STRIDE analysis

ID	ASSET / FLOW	S	T	R	I	D	E
D1	Customer data	X			X		
D2	Orders data	X			X		
D3	Products data						
D4	Auth tokens	X	X		X		
D5	App logs				X		
D6	Payment data	X			X		
D7	Secrets & Keys		X	X	X	X	X
C1	Frontend					X	
C2	Backend	X	X	X	X	X	X
C3	Databases	X	X		X	X	
C4	REDIS cache						
C5	Authentication API	X					X
C6	Payment API	X			X	X	
C7	CDN and Storage server		X	X	X	X	
C8	Reverse proxy & WAF			X		X	
F1	Authentication flow					X	
F2	Payment flow		X			X	
F3	Cart flow		X			X	
F4	Orders flow					X	
F5	Catalog flow					X	
F6	Administrative flow		X			X	X
F7	Logs flow						
A1	Customer		X		X		
A2	Administrator	X	X	X	X	X	X
A3	Payment third-party			X	X	X	
A4	Internal services (jobs, crons)		X	X		X	X