

Resilient SADAD Payment Network — EMAM Framework Report

Author: Date:

(Understand) — Why Resilience > Security, Threat Analysis

- **Saudi context**
 - National payments like SADAD, mada, and SARIE are critical to economy and daily life. Outages impact commerce, government services, and public trust.
 - Historical threats such as Shamoon-class malware demonstrated destructive potential to erase disks and disrupt operations.
 - Regulatory oversight by SAMA emphasizes continuity of critical financial services (BCP/DR, resilience testing, incident reporting).
 - **Why resilience > security**
 - Security seeks to prevent compromise; resilience assumes incidents will occur and ensures services continue meeting minimum objectives (SLOs) while recovering quickly.
 - Business impact is measured by RTO/RPO and sustained service levels under failure, not just breach prevention.
 - Design for failure: graceful degradation, fallback paths, and rapid recovery reduce societal harm.
 - **Islamic principles**
 - (Amanah): Stewardship of citizens' data and funds.
 - (Adl): Fair access and equitable service continuity.
 - (No harm): Minimize harm via rapid containment, transparent comms, quick restoration.
 - **Threat model (STRIDE + operational)**
 - Actors: state-level adversaries, cybercriminals, insider threats, supply-chain compromises, cloud zone/regional failures, network partitions, DDoS, destructive malware (e.g., disk wipers), misconfiguration.
 - Assets: payment API, transaction ledger, settlement services, KMS/keys, customer PII, observability & CI/CD, infra as code.
 - Attack paths: credential theft (privilege escalation), lateral movement to ops hosts, CI secrets exfiltration, container breakout, malware spread via software updates, API abuse, data corruption.
 - Environmental: AZ failure, DNS outage, ISP peering issues, data center incidents.
-

(Practice) — 4 Rs and Swiss Cheese Model

- **4 Rs of resilience**
 - Robustness: Safe defaults, idempotent operations, circuit breakers, rate limits, schema evolution, strong typing, immutability.
 - Redundancy: Multi-AZ replicas, active-active instances, backup KMS, replicated object storage, warm DR region, redundant network paths.
 - Resourcefulness: Runbooks, automated failover, feature flags, chaos drills, emergency comms, break-glass access with approvals.
 - Rapidity: Automated detection (Prometheus alerts), canary rollback, IaC re-provisioning, snapshot restore, pre-approved playbooks.
 - **Swiss Cheese layers**
 - Layer 1: Preventive controls (CI scanning, SBOM, image signing, least privilege IAM).
 - Layer 2: Detect/observe (Prometheus, logs, tracing, SLOs, blackbox probes).
 - Layer 3: Contain/limit blast radius (network policies, microsegmentation, KMS key scoping, namespace isolation).
 - Layer 4: Recover/continue (multi-AZ, backups, DR automation, replay from ledger, idempotency keys).
 - Layer 5: Governance (change mgmt, audits, postmortems, chaos practice cadence).
 - **Practical chaos scenarios (see `chaos-tests/`)**
 - Pod kill, node drain, network partition, AZ outage.
 - DDoS traffic ramp.
 - Shamoon-like disk-wipe simulation (dry-run) + health validation.
-

(Master) — Resilient Architecture & Critical Services

- **Service overview**
 - Payment API (Spring Boot), idempotent processing pipeline, async ledger writer, settlement connector, observability sidecar.
 - Kubernetes with HPA, PDB, anti-affinity, topology spread; Prometheus Operator ServiceMonitor & PrometheusRule for SLOs.
 - Chaos Monkey profile for in-app faults; external chaos scenarios.
 - Terraform for reproducible infra (expand with EKS/VPC modules).
- **Architecture (Mermaid)**

```
flowchart LR
    subgraph Region A
        subgraph AZ1
            A1[Payment Pod]:::app --> RDS1[(Ledger/DB)]
        end
        subgraph AZ2
```

```

    A2[Payment Pod]:::app --> RDS2[(Ledger/DB)]
  end
end
subgraph Region B (DR)
  B1[Payment Pod (warm)]:::app --> RDSB[(Replica/Backup)]
end
Client-->LB[Ingress/Service]
LB-->A1
LB-->A2
Prom[Prometheus]-.->A1
Prom-.->A2
classDef app fill:#e6f7ff,stroke:#007acc

```

- **Patterns**

- Idempotency keys; outbox pattern for reliable ledger writes.
- Circuit breakers + retries with jitter.
- Blue/green or canary deployments.
- Secrets from KMS; short-lived credentials.
- Backups with point-in-time recovery; restore drills.

- **Kubernetes configs**

- `kubernetes/deployment-tier1.yaml`: readiness/liveness, resource requests/limits, anti-affinity, topology spreads, PDB.
- `kubernetes/service-monitor.yaml`: metrics scrape + `Service`.
- `kubernetes/prometheus-rules.yaml`: SLO alerts for availability, latency p95, error rate, saturation.

(Excellence) — Innovation & Vision 2030

- **Innovation**

- Self-healing with policy-as-code to auto-quarantine compromised pods.
- Proactive capacity predictions using telemetry.
- Immutable infra with rapid rehydration from IaC.

- **Vision 2030 alignment**

- Enable fintech ecosystem and cashless society goals with high uptime and trust.
 - Data residency & compliance by design; transparency and public confidence.
 - Ethical stewardship grounded in Islamic principles of fairness and harm minimization.
-

Recovery Procedures (Runbooks)

See `docs/runbooks/recovery-procedures.md` for malware containment, DR failover, and key rotation playbooks, including RTO/RPO targets and verification steps.

Chaos, Monitoring, and SLOs

- Run app with Chaos Monkey profile (local):
 - `mvn spring-boot:run -Dspring-boot.run.profiles=chaos`
 - `scripts/chaos-monkey-demo.sh`
 - Kubernetes monitoring:
 - Apply `kubernetes/service-monitor.yaml` and `kubernetes/prometheus-rules.yaml` with kube-prometheus-stack.
 - Scenarios:
 - See `chaos-tests/` YAMLS for pod kill, partition, zone outage, DDoS, and Shamoon simulation.
-

Verification Results

Application Build & Tests

Test Phase	Status	Details
Maven Build	SUCCESS	Compiled 3 source files
Resources	SUCCESS	Copied 2 resources to target/classes
Unit Tests	SUCCESS	All tests passed
Build Time	1.045s	Fast build cycle
Date	Verified	2025-11-26T20:03:15+03:00

Command: `mvn clean test`

Chaos Test Scenarios

Scenario	File	Severity	Status	Description
AZ Outage	<code>az-outage-simulation.yaml</code>	Critical	Ready	Simulates complete Availability Zone failure

Scenario	File	Severity	Status	Description
DDoS (Eid)	ddos-eid-scenario.yml	High	Ready	High-traffic scenario during Eid period
Network Partition	network-partition-scenario.yml	High	Ready	Simulates network split between zones
Pod Kill	pod-kill-scenario.yml	Medium	Ready	Kills payment pod, verifies availability
Shamoon Malware	shamoon-simulation.yml	Critical	Ready	Disk-wipe simulation with recovery

Command: `bash chaos-tests/run-all-tests.sh`

Total Scenarios: 5 (2 Critical, 2 High, 1 Medium)

Infrastructure Components

Component	Technology	Configuration	Resilience Features
VPC	AWS VPC	10.0.0.0/16	3 Availability Zones
Network	Public/Private Subnets	3 public + 3 private	Multi-AZ isolation
NAT	NAT Gateway	One per AZ	High availability
Compute	EKS 1.27	Auto Scaling	Min: 3, Max: 6 nodes
Nodes	t3.medium	ON_DEMAND	Spread across AZs
Monitoring	Prometheus	ServiceMonitor	SLO-based alerts
Orchestration	Kubernetes	HPA + PDB	Self-healing

Terraform Files: `main.tf`, `variables.tf`, `outputs.tf`, `versions.tf`

Submission

- Repository: include this EMAM report, runbooks, chaos scenarios, Terraform, and Kubernetes manifests.
- PDF: export this document via `scripts/build-pdf.sh` (requires pandoc) or Print-to-PDF.
- Checklist:
 - EMAM report covers Saudi context and Islamic principles.
 - 4 Rs and Swiss Cheese applied with practical tests.

- Architecture, SLOs, and recovery procedures included.
- Chaos tests runnable (stubs or integrated) and monitored.