Cybersecurity Testing & Security Implementation Summary

SMART LOSS CONTROL – An AI-Integrated Retail Inventory Web Application
Environment: Local Testing (Kali Linux VM)
Prepared by: Cybersecurity Team – **GROUP 70**

---

1. Why We Conducted This Security Testing

For the final phase of our Smart Loss Control website, my team and I wanted to ensure that our backend API and overall application were not just functional, but truly secure.

We focused on:

- Confirming authentication was properly enforced
- Validating access control mechanisms
- Checking for common API misconfigurations
- Reviewing encryption considerations
- Aligning with NDPR data protection principles
- Identifying areas for improvement before final submission

All testing was conducted in a controlled local environment using Kali Linux.

---

2. How We Tested the Application

To keep the assessment structured and practical, we divided our testing into clear phases and used industry-recognized tools.

Tools We Used:

- **Nmap** – To discover active services and open ports
- **Netstat** – To confirm port exposure
- **Curl** – To inspect HTTP headers and test endpoints
- **Gobuster** – To check for hidden or unintended endpoints
- **Swagger UI** – To manually review and test API structure
- **Browser-based testing** – To validate authentication behavior

- This approach allowed us to combine automated checks with hands-on validation.

---

3. What We Tested & What We Found

3.1 Service & Port Verification

We first confirmed that the backend was running only on the intended port and that no unnecessary services were exposed.

Result:
We did not detect any abnormal port exposure.

This tells us the application maintains a minimal attack surface from open services.

3.2 Endpoint Enumeration

Using Gobuster and Swagger, we:

- Reviewed all available endpoints
- Checked for undocumented or hidden routes
- Ensured the API structure matched intended functionality

Result:
No unexpected endpoints were discovered.

This gave us confidence that the backend was not exposing unintended functionality.

3.3 HTTP Security Headers

We inspected the application's HTTP security headers and confirmed that Helmet middleware was properly enabled.

The following headers were configured:

- Content-Security-Policy (CSP)
- HTTP Strict Transport Security (HSTS)
- X-Frame-Options
- X-Content-Type-Options

These configurations help protect against common browser-based attacks such as clickjacking and MIME-type sniffing.

Status: Secure configuration confirmed.

3.4 CORS Policy Review

During our review, we observed that:

Access-Control-Allow-Origin was set to *.

This means any external domain could technically interact with the API.

Risk Level: Medium (if deployed publicly)

Our Recommendation:
Before production deployment, CORS should be restricted to only the trusted frontend domain.

3.5 Public Health Endpoint

We noticed that the /health endpoint was accessible without authentication.

While this is common for monitoring purposes, it should only return minimal system information.

Risk Level: Low

Our Recommendation:
Limit the response to something simple like status: OK to avoid exposing system details.

3.6 Authentication & Access Control Testing

Since authentication was implemented by the backend team, we intentionally tested:

- Access to protected endpoints without a token
- How the system responded to unauthorized requests
- Basic role enforcement behavior

Results:

- Protected endpoints returned HTTP 401 Unauthorized when accessed without valid authentication

- Token enforcement worked correctly
- Unauthorized access attempts were successfully blocked

From our testing, we confirmed that authentication is properly enforced at a foundational level.

---

4. QR Code & HTTPS Considerations

During local testing, the QR feature generated HTTP links because we were in a development environment.

Mobile browsers flagged these links as “Not Secure,” which is expected in local testing.

However, for real-world deployment:

TLS (HTTPS) must be enforced to prevent:

- Token interception
- Session hijacking
- Man-in-the-middle attacks

We recommended secure tunneling during development to simulate HTTPS behavior and proper TLS enforcement in production.

---

5. Encryption & Data Protection Considerations

AES-256 Encryption

The system design incorporates AES-256 encryption for handling sensitive data.

This is an industry-standard encryption algorithm that provides strong protection for stored sensitive information, including transactional and user-related data.

NDPR Alignment

We intentionally aligned the application with key Nigeria Data Protection Regulation (NDPR) principles by ensuring:

- Data minimization practices
- Controlled access to sensitive endpoints

- Authentication enforcement
- Encryption considerations
- Protection of transactional records

This shows that compliance was considered as part of the system design, not as an afterthought.

---

6. Overall Security Assessment

Security Posture: Moderate to Strong

Key Highlights:

- No critical vulnerabilities were identified during testing
- Authentication is properly enforced
- Security headers are correctly configured
- Encryption standards are incorporated
- Compliance awareness is evident
- Only minor configuration improvements are required

Overall, we believe the application demonstrates intentional security design rather than reactive security implementation.

---

7. Recommended Improvements Before Production

Before real-world deployment, we recommend:

1. Restricting CORS to trusted domains
2. Enforcing HTTPS (TLS)
3. Implementing rate limiting on authentication endpoints
4. Ensuring QR tokens are short-lived and single-use
5. Enabling structured audit logging
6. Conducting a final penetration test

---

8. Closing Note

From a cybersecurity perspective, my team and I are confident that the application has a strong foundational security structure.

Security controls were validated, authentication was enforced correctly, encryption standards were considered, and compliance alignment was integrated into the design process.

While a few improvements are necessary for full production readiness, no critical security flaws were identified during our testing phase.