Kintsugi Kode

Case Study - Legacy E-commerce Store on MODX

Legacy e-commerce store on MODX: 3 critical risks closed in one day

The system worked. Revenue came in. Pages loaded. Nobody knew what was hiding inside.

The store looked fine from the outside. Inside, we found unsupported PHP, public server logs, broken cron jobs and exposed CMS files.

The situation

A live online store selling ATV and snowmobile accessories to customers in the US, Canada and Europe. MODX Revolution, 40+ components, running on the same server for years. No documentation. No staging environment. The original integrations were set up by a developer who was no longer involved.

The owner knew the system worked. But every change felt risky.

Area	Details
System type	E-commerce store for ATV/UTV and snowmobile accessories
Platform	MODX Revolution + miniShop2 + 40+ additional components
Hosting	Hetzner Cloud server
Markets	US, Canada, Europe
Audit type	Kintsugi Kode - 72-hour Legacy System Audit
Discovery mode	Read-only discovery first; production changes only after review and approval

What we found

Finding	Severity	Business impact
PHP 7.4 in production	Critical	End-of-life since November 2022. No security patches for more than two years.
Server log archive in public folder	Critical	A 13MB archive with internal server data was downloadable by direct URL.
3 cron jobs calling missing scripts	Critical	Automated tasks fired every 10 minutes and failed silently. A legacy integration had stopped working at an unknown point.
CMS source files exposed publicly	High	MODX installation/source files were present in the public directory, exposing system structure.
40+ components untouched since 2021	High	Payments, cart, search and captcha-related components were years out of date.
Online card payments unavailable	High	Buyers discovered payment limitations too late in the purchase flow.

What changed the same day

The discovery audit did not modify production. After review and approval, three critical risks were closed the same day:

Risk	Action taken
Public server log archive	Removed from the public web directory. Internal log data was no longer exposed.
CMS source files exposed publicly	Removed from the public directory. Attack surface reduced.
Broken automated cron jobs	Documented and disabled. The legacy integration was confirmed as obsolete.

3	1 day	0
critical risks closed	from discovery to first fixes	production changes during discovery

What the owner received

- Full system map
- Risk register
- Technical debt review
- Quick wins list
- 30/60/90-day modernization plan

What happens next

The remaining risks became a staged modernization roadmap. No blind rewrite. No random fixes. First stabilize, then upgrade, then migrate where it makes sense.

Phase	Action
Stabilize	Close quick wins, document integrations, confirm backups, remove obsolete public files and broken automation.
Update through staging	Build a staging environment and update critical components in order: payments, cart, captcha, search and operational modules.
Modernize safely	Upgrade PHP to a supported version, restore card payments and define the longer-term platform roadmap.

Find out what is hiding inside your legacy system.
Kintsugi Kode maps old websites, CRMs and backend systems before something breaks, before migration, before a blind rewrite.

cases@hlinor.com | hlinor.com/kintsugi-kode