CYBERCILE

For FinTech & Payment Leaders

Payment System Security Validation Blueprint

What Teams Must Validate From Design to Continuous Assurance

Build securely. Test what matters. Produce evidence
buyers, auditors, customers, and sponsor banks can use.

For FinTechs, payment platforms, MSBs, remittance companies, digital wallets, and financial institutions.

CyberCile.com

WHY THIS MATTERS

A secure product is not the same as a passed scan

Payment platforms concentrate identity, authorization, sensitive data, third-party integrations, and irreversible business actions. A scanner can find exposed software weaknesses. It cannot reliably prove that a user cannot change a recipient, replay a payout, cross a tenant boundary, abuse a refund, or misuse an administrative workflow.

This blueprint combines secure software practices with payment-specific validation. It is designed to produce two outcomes at the same time: lower exploitability and stronger evidence for commercial and compliance reviews.

What makes the process evidence-based

- **NIST SSDF** integrates secure practices into the software lifecycle to reduce vulnerabilities, reduce their potential impact, and prevent recurrence.
- **OWASP ASVS** provides testable application-security requirements; OWASP API Security highlights risks such as broken object authorization, misconfiguration, inventory gaps, and unsafe third-party API consumption.
- **PCI DSS** provides a baseline of technical and operational requirements for protecting payment account data when it is in scope.
- **CISA Secure by Design** reinforces ownership, secure defaults, and eliminating preventable vulnerability classes at the source.

How to use this blueprint

1. Pick the money flow

Start with one critical journey: onboarding, funding, transfer, payout, refund, settlement, or privileged account change.

2. Follow the five phases

Complete the defined work and decision gate in each phase. Do not treat the activities as a one-time checklist.

3. Keep the evidence

Store the artifacts listed in the evidence pack. Evidence should show scope, decision, owner, date, and verification status.

4. Revalidate on change

Repeat targeted testing after material code, architecture, identity, vendor, cloud, or transaction-workflow changes.

DECISION GATE **No “proven process” guarantees security. This process is proven in the practical sense that it applies recognized controls, independent testing, remediation verification, and continuous feedback.**

OPERATING MODEL

Five principles for companies that move money

01

Protect money movement first

Map how value enters, changes state, moves, reverses, settles, and leaves. Give the highest assurance to the paths that can create financial loss or an incorrect ledger state.

02

Treat every trust boundary as testable

Test customer-to-app, app-to-API, tenant-to-tenant, staff-to-admin, platform-to-vendor, webhook, sponsor-bank, and cloud boundaries.

03

Make risky actions hard and visible

Use least privilege, strong authentication, step-up controls, approvals, velocity limits, tamper-evident logging, and clear ownership.

04

Validate beyond the scanner

Combine automation with manual business-logic testing and code review. A clean scan does not prove correct authorization, state, sequencing, or abuse resistance.

05

Design for containment and recovery

Use circuit breakers, transaction limits, key rotation, safe rollback, incident playbooks, and verified restoration paths before an incident occurs.

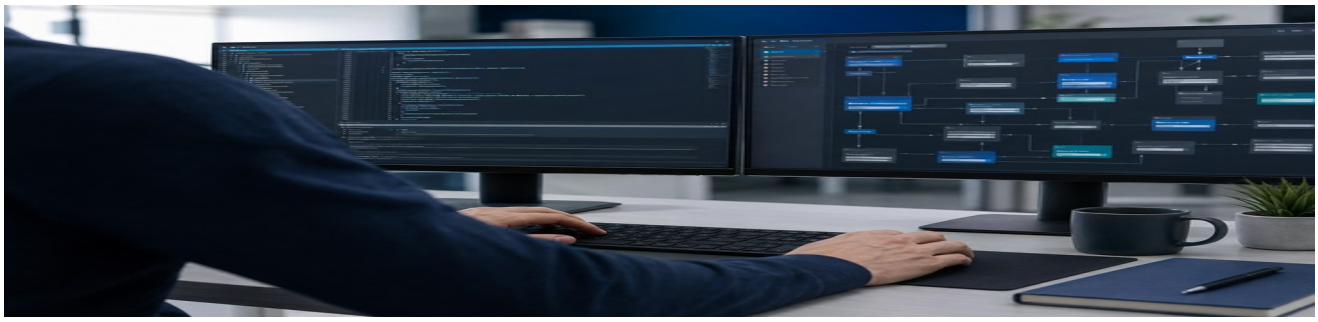
The control loop

UNDERSTAND > DESIGN > BUILD > ATTACK > FIX > VERIFY > MONITOR > IMPROVE

The loop is complete only when a fix is independently verified and the lesson changes the requirement, test, control, or monitoring rule that allowed the issue to exist.

A Understand and Design

Outcome: a testable threat model for the real payment flow



Work to complete

- Map the transaction and data flow from initiation through authorization, posting, settlement, reversal, reconciliation, and reporting.
- Inventory crown jewels: ledger state, payment instructions, identity proof, secrets and keys, PII, cardholder data, privileged functions, and evidence logs.
- Mark trust boundaries and dependencies, including sponsor banks, processors, KYC providers, webhook consumers, cloud services, and support tools.
- Write abuse cases alongside functional requirements. Each abuse case needs an owner, expected control, test method, and acceptance criterion.
- Define transaction limits, velocity controls, approval rules, step-up authentication, reconciliation tolerances, and emergency stop authority.
- Separate cyber, fraud, compliance, engineering, and operations ownership. Define who decides, who implements, who verifies, and who accepts residual risk.

Required artifacts

Money-flow diagram

Systems, actors, data, transaction states, third parties, and trust boundaries.

Threat and abuse-case register

Scenario, precondition, business impact, control, owner, test, and status.

Security acceptance criteria

Specific, testable requirements attached to the product backlog.

Risk decision record

Named approver, rationale, compensating control, expiry date, and revalidation trigger.

DECISION GATE **Architecture and product owners approve the flow, threats, controls, and acceptance criteria before high-risk development begins.**

B Build and Integrate Securely

Outcome: protected code, dependencies, pipelines, secrets, and configurations

Controls for high-risk payment code

- Enforce authorization on the server for every object and action. Deny by default. Never trust a client-supplied role, tenant, balance, limit, or transaction state.
- Make transfer, payout, refund, and reversal operations idempotent. Bind requests to the authenticated subject, intended recipient, amount, currency, and valid state transition.
- Protect webhooks with strong signing, freshness checks, replay protection, strict schemas, safe retries, and least-privileged destinations.
- Use approved secrets and key management. Keep credentials out of source, images, logs, build output, and client-side code. Rotate on schedule and on suspicion.
- Review high-risk code with a second qualified reviewer. Require explicit review for authorization, ledger updates, transaction-state logic, administrative actions, and cryptography.
- Run static analysis, dependency analysis, secret scanning, and infrastructure checks in CI. Fail the pipeline on defined conditions, not on an unowned warning backlog.
- Generate and retain a component inventory or SBOM where useful. Pin versions and define a response path for vulnerable dependencies.
- Protect the build and release path with branch rules, isolated runners, signed or traceable artifacts, restricted deployment rights, and separation of duties.

Developer definition of done

- ☐ The abuse case has an automated negative test.
- ☐ Authentication, authorization, and tenant isolation are verified at the service boundary.
- ☐ Sensitive events are logged without leaking sensitive values.
- ☐ Failure behavior is safe, deterministic, observable, and recoverable.
- ☐ The change is traceable from requirement to commit, test result, build artifact, and deployment.

DECISION GATE **No unresolved critical design weakness. Every privileged or money-moving change has qualified review and passing negative tests.**

C Attack, Review, and Verify

Outcome: evidence of what is exploitable, what was fixed, and what remains

Use layers of testing

Automated assurance

SAST, SCA, secret scanning, IaC checks, API schema tests, DAST, and authenticated vulnerability scanning.

Manual penetration testing

Web, API, mobile, cloud, and identity testing with business context and written authorization.

Secure code review

Line-level and data-flow review of authorization, ledger logic, transaction state, cryptography, and dangerous integrations.

Fix verification

Focused retesting that reproduces the original path, confirms the control, checks for bypasses, and records proof.

Test like a payment attacker

- Change amount, currency, recipient, fee, funding source, transaction ID, account, tenant, status, and approval data at every boundary.
- Replay, duplicate, reorder, race, cancel, refund, reverse, and partially complete transaction workflows.
- Move across roles and tenants. Test customer, merchant, partner, support, operations, developer, and administrator paths.
- Abuse password reset, MFA recovery, device enrollment, support-assisted changes, and session lifecycle.
- Spoof, replay, delay, and alter webhooks and third-party responses. Test fail-open behavior and unsafe trust in vendors.
- Test cloud exposure, SSRF, secrets, metadata access, storage permissions, deployment drift, and excessive service identity permissions.

Severity needs business context

Prioritize by exploitability plus business consequence: unauthorized value movement, incorrect ledger state, cross-tenant access, sensitive-data exposure, privileged control, regulatory impact, detection gap, and blast radius. CVSS can inform severity, but it should not replace payment-specific impact analysis.

DECISION GATE **Critical and high findings are fixed and verified, or formally accepted by an authorized risk owner with a time limit and compensating controls.**

D Launch with Controlled Exposure

Outcome: the assessed build launches with limits, visibility, and recovery options

- Confirm the deployed artifact, configuration, feature flags, infrastructure, and environment match the assessed scope.
- Validate production access, secrets, signing keys, service identities, allowlists, and administrative paths.
- Enable transaction limits, velocity controls, step-up checks, dual approvals, circuit breakers, and safe rollback where appropriate.
- Confirm logs and alerts cover failed authentication, privilege changes, recipient changes, transaction anomalies, overrides, webhooks, key use, and emergency controls.
- Run a go-live exercise: trigger an alert, pause or constrain a risky action, locate evidence, notify owners, and restore safely.

DECISION GATE **The release owner signs a launch evidence pack with verified build identity, residual risk, monitoring, limits, rollback, and incident contacts.**

E Operate and Continuously Revalidate

Outcome: assurance stays current as the system and threat surface change

- Continuously monitor the external attack surface, exposed services, new assets, certificate and DNS changes, cloud drift, and high-risk vulnerabilities.
- Triage findings using reachability, exploitability, privilege, transaction impact, data sensitivity, and evidence quality.
- Trigger targeted testing after material changes to authorization, money flow, APIs, webhooks, cloud, vendors, keys, identity, or administrative workflows.
- Retest remediated critical and high findings. Track repeat findings to the missing root-cause control.
- Exercise incident and recovery playbooks. Review access, keys, dependencies, and third-party boundaries on a defined cadence.

DECISION GATE **Security, engineering, and compliance review assurance metrics and overdue risk decisions at least quarterly.**

PAYMENT-SPECIFIC VALIDATION

Twelve scenarios your test plan must cover

Account takeover

MFA bypass, reset abuse, session theft, device enrollment, support-assisted recovery.

Transaction tampering

Change amount, currency, fees, recipient, payout account, funding source, limits, or approval state.

Race and state errors

Create inconsistent balances or approvals through concurrent, reordered, partial, or timed operations.

Webhook and partner trust

Spoof source, bypass signature or freshness checks, replay delivery, or poison trusted partner data.

API inventory and drift

Find old versions, shadow endpoints, debug routes, weak schemas, or inconsistent authorization.

Sensitive-data exposure

Leak PII, account data, tokens, authentication data, statements, reports, or secrets through APIs and logs.

Object and tenant authorization

Access another customer, merchant, account, transaction, statement, file, or administrative object.

Replay and duplication

Repeat signed requests, webhooks, idempotency keys, payout instructions, or asynchronous jobs.

Refund and reversal abuse

Refund beyond allowed amount, reverse another party's payment, or exploit reconciliation gaps.

Admin and support abuse

Escalate privilege, bypass approvals, change recipient or identity data, or hide actions from audit.

Cloud and secret exposure

Reach metadata, steal credentials, abuse service identities, or access sensitive storage and logs.

Detection and containment

Perform high-risk actions without alerts, exceed limits, or defeat pause, rollback, and key-rotation procedures.

A useful test case has six parts

Precondition | Attacker action | Expected control | Expected evidence | Business impact | Pass or fail decision

DECISION GATE **Every critical money-moving flow has at least one positive test, one negative test, one abuse case, and one observable alert or evidence path.**

LEAN-TEAM IMPLEMENTATION

A practical 90-day rollout

Start with one high-value payment journey. Expand only after the first loop produces verified fixes and reusable evidence.

Window	Work	Proof produced	Exit condition
Days 1-30	Map the payment flow, assets, owners, dependencies, threats, controls, and top abuse cases. Baseline exposure and open findings.	Flow diagram, asset inventory, threat register, decision log, initial scan and manual spot checks.	Top five loss scenarios have owners, controls, tests, and dates.
Days 31-60	Fix and independently retest the highest-risk weaknesses. Add logging, limits, approvals, key controls, and negative tests.	Verified retest notes, code-review record, updated tests, evidence samples, residual-risk decisions.	Critical paths cannot be bypassed using the known scenarios.
Days 61-90	Operationalize change triggers, continuous validation, metrics, access and key reviews, and an incident exercise.	Assurance calendar, dashboard, runbook, tabletop record, evidence pack, next-quarter plan.	The loop has owners, cadence, escalation, and executive review.

Who owns what

Executive or risk owner

Sets tolerance, funds remediation, accepts time-bound residual risk.

Product and engineering

Own secure requirements, implementation, tests, release identity, and fixes.

Security

Owns threat modeling, test strategy, independent validation, triage, and assurance reporting.

Compliance and operations

Map evidence to obligations, operate controls, reconcile exceptions, and exercise response.

DECISION GATE **Day 90 is not completion. It is the point at which the first repeatable assurance loop is operating.**

CONTINUOUS ASSURANCE

Minimum viable validation cadence

When	Minimum action	Owner evidence
Every high-risk release	Peer review, automated scans, negative abuse-case tests, deployment identity, and change-trigger decision.	Pull request, test run, release record
Continuously	External exposure and vulnerability signal monitoring with owned triage.	Ticket and response timeline
Monthly	Review new critical and high findings, overdue fixes, asset coverage, and material changes.	Assurance review record
Quarterly	Access, keys, third parties, attack surface, recurring findings, and targeted manual validation.	Quarterly evidence pack
Annually or major change	Independent penetration test of relevant web, API, mobile, cloud, and identity scope.	Report and management response
After remediation	Focused retest of critical and high findings plus obvious bypass paths.	Verification record

Metrics that drive decisions

Critical-path coverage

% of critical money flows with security requirements, abuse cases, tests, and evidence.

Time to triage

Median and 90th percentile time to own critical and high findings.

Verified closure rate

% of critical and high fixes independently retested and closed.

Repeat-finding rate

% of findings caused by a previously known weakness or missing systemic control.

Change-trigger compliance

% of material changes that received the required targeted revalidation.

Asset coverage

% of in-scope external assets, APIs, cloud accounts, and applications monitored and tested.

Avoid vanity metrics such as raw vulnerability counts without scope, severity quality, exposure, ownership, or business impact.

QUICK REFERENCE

Payment security validation checklist

Governance and evidence

- ☐ Critical payment journeys and crown jewels are named and owned.
- ☐ Security acceptance criteria and abuse cases are in the product backlog.
- ☐ Residual-risk decisions name an approver, expiry date, and compensating control.
- ☐ Evidence is retained with scope, date, owner, result, and version.

Architecture and trust

- ☐ Money and data flows include transaction states, trust boundaries, third parties, and failure paths.
- ☐ Least privilege, approvals, limits, reconciliation, and emergency controls are defined.
- ☐ Tenant, partner, webhook, cloud, and administrative boundaries are explicitly tested.

Secure build

- ☐ High-risk code receives qualified review and negative tests.
- ☐ Authorization, idempotency, replay resistance, state transitions, and failure behavior are verified.
- ☐ Secrets, dependencies, build pipelines, and deployment permissions are protected and traceable.

Independent validation

- ☐ Automated findings are triaged and owned.
- ☐ Manual penetration testing covers business logic, APIs, identity, cloud, and payment abuse.
- ☐ Secure code review covers high-risk money-moving and privileged logic.
- ☐ Critical and high fixes are retested with proof.

Operations and response

- ☐ Deployed build and configuration match the assessed scope.
- ☐ High-risk actions produce useful alerts and tamper-resistant evidence.
- ☐ Incident, pause, rollback, key rotation, and recovery procedures are exercised.
- ☐ Material changes trigger targeted revalidation and quarterly assurance review.

MAKE ASSURANCE USABLE

The evidence pack buyers and auditors can follow

A useful evidence pack is not a folder of screenshots. It tells a traceable story from requirement to test to decision.

01 Scope and system identity Applications, APIs, cloud environments, versions, commit or build identifiers, exclusions, dates, and owners.

02 Architecture and threat record Payment flows, trust boundaries, crown jewels, abuse cases, controls, and assumptions.

03 Test plan and execution Methods, roles, accounts, environments, coverage, limitations, and actual results.

04 Findings and business impact Reproduction, affected flow, evidence, exploitability, impact, severity rationale, and owner.

05 Remediation and verification Fix reference, reviewer, deployment identity, retest method, result, date, and remaining risk.

06 Management decision Accepted risk, compensating controls, deadline, accountable approver, and revalidation trigger.

07 Operational assurance Monitoring coverage, access and key reviews, incident exercises, cadence, and metrics.

Definition of verified

A finding is verified closed when an independent tester can no longer reproduce the original path, reasonable bypasses fail, the expected control and evidence are present, and the tested build or environment is identified. “Developer says fixed” is a status update, not verification.

DECISION GATE **Use this evidence pack as the reusable output of every assessment, retest, material change, and quarterly assurance review.**

WHEN INDEPENDENT PROOF MATTERS

Turn the blueprint into verified assurance

This guide helps a lean team organize the work. It does not prove that a live payment system is secure. Proof requires testing the real system under written authorization, verifying remediation, and repeating validation as the system changes.

Penetration Testing

Manual web, API, mobile, cloud, identity, and business-logic validation for payment environments.

Secure Code Review

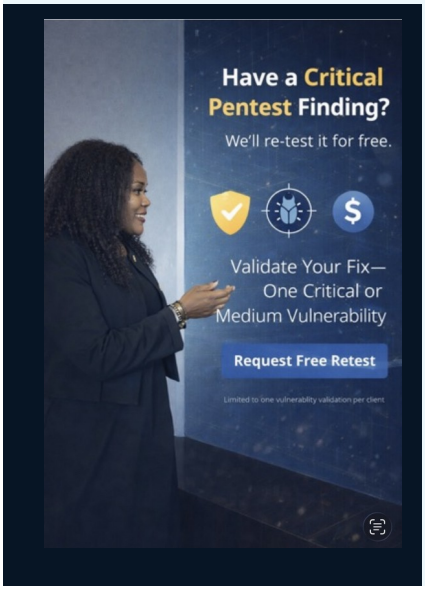
Focused review of authorization, ledger and transaction-state logic, privileged operations, and risky integrations.

Continuous Validation

Ongoing attack-surface review, targeted testing, remediation guidance, retesting, and evidence support.

Free Finding Verification

CyberCile will independently retest one qualifying critical or high finding from a prior assessment, subject to scope and written authorization.



CYBERCILE

Security validation for companies that move money.

Founded by Cecile Mengue, CyberCile brings offensive-security experience in payment environments to focused assessments, code review, remediation verification, and continuous assurance.

Visit CyberCile.com | support@cybercile.com

Standards and guidance used

NIST Secure Software Development Framework, SP 800-218

OWASP Application Security Verification Standard

OWASP API Security Project

PCI Data Security Standard overview

CISA Secure by Design

This blueprint is educational guidance. It is not legal advice, a compliance assessment, or a guarantee against compromise. Requirements and scope should be tailored to your technology, contracts, risk tolerance, and applicable obligations.