Online Payments Processing Platform

Owner: A development team
Reviewer: A security architect
Contributors: development engineers, product managers, security architects
Date Generated: Tue Oct 07 2025

Executive Summary

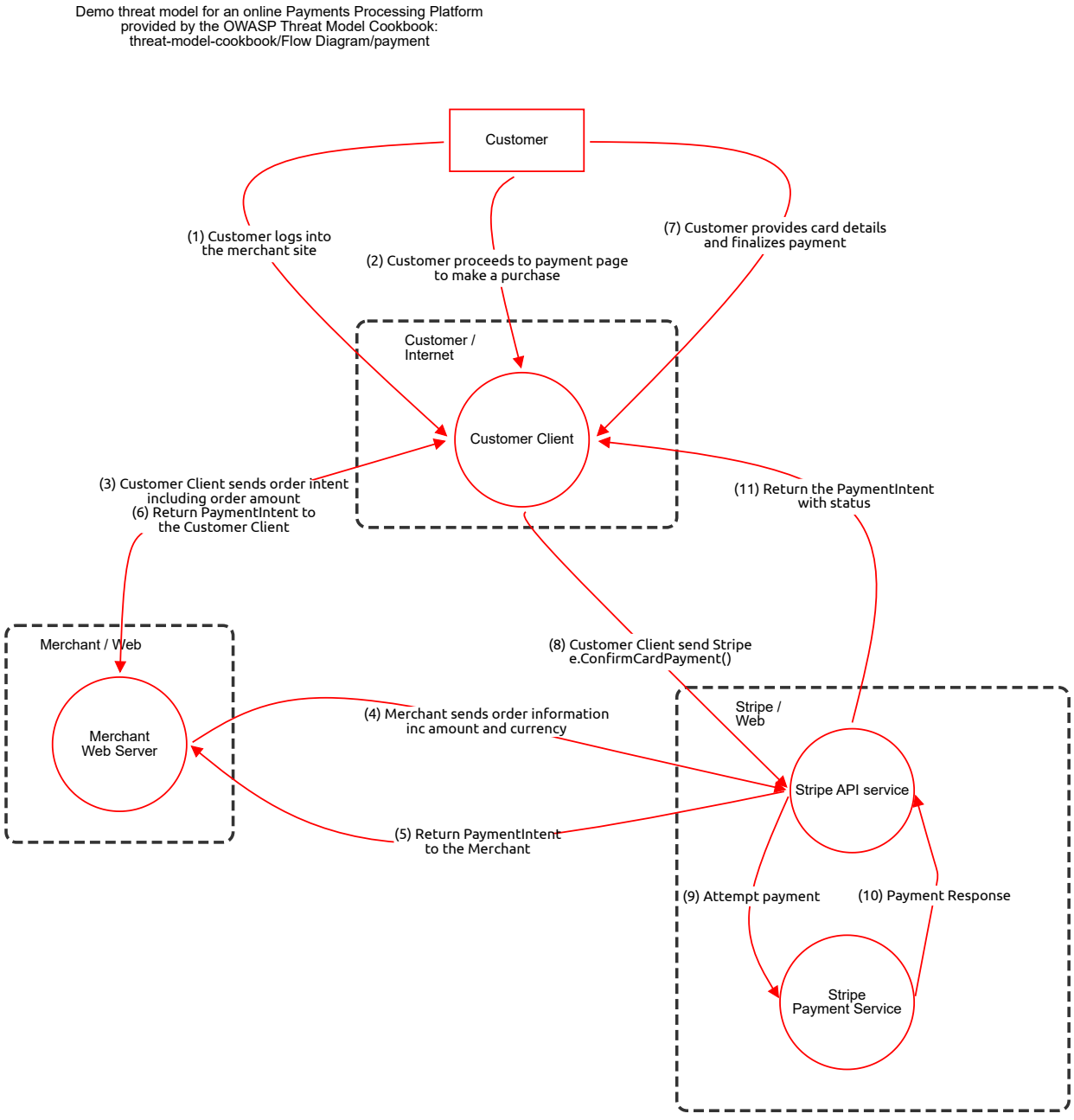
High level system description

This threat model has been provided by the OWASP Threat Model Cookbook:
threat-model-cookbook/Flow Diagram/payment

Summary

Total Threats	26
Total Mitigated	0
Total Open	26
Open / Critical Severity	0
Open / High Severity	17
Open / Medium Severity	9
Open / Low Severity	0

Payment


Demo threat model for an online Payments Processing Platform
provided by the OWASP Threat Model Cookbook:
threat-model-cookbook/Flow Diagram/payment
Customer
(1) Customer logs into
the merchant site
(2) Customer proceeds to payment page
to make a purchase
(7) Customer provides card details
and finalizes payment
Customer /
Internet
Customer Client
(3) Customer Client sends order intent
including order amount
(6) Return PaymentIntent to
the Customer Client
(11) Return the PaymentIntent
with status
Merchant / Web
Merchant
Web Server
(8) Customer Client send Stripe
e.ConfirmCardPayment()
(4) Merchant sends order information
inc amount and currency
(5) Return PaymentIntent
to the Merchant
Stripe /
Web
Stripe API service
(9) Attempt payment
(10) Payment Response
Stripe
Payment Service

Payment

Customer (Actor)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing of Customer Identity	Spoofing	High	Open		The Customer actor initiates authentication flows to the Customer Client process across trust boundaries. Without strong authentication mechanisms, an attacker could impersonate legitimate customers to access payment services and initiate fraudulent transactions.	- Implement multi-factor authentication (MFA) - Use strong password policies - Implement account lockout mechanisms - Monitor for suspicious login patterns

Customer Client (Process)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	XSS Attack on Customer Client	Tampering	High	Open		The Customer Client process handles sensitive payment data and communicates across multiple trust boundaries (Customer/Internet, Merchant/Web, and Stripe/Web zones). Client-side code could be tampered with through XSS attacks to steal payment information or modify transaction details.	- Implement Content Security Policy (CSP) - Sanitize all user inputs - Use secure coding practices - Regular security scanning of client-side code
	Man-in-the-Browser Attack	Information Disclosure	High	Open		The Customer Client process handles credit card details and payment information within the browser environment. Malware or browser extensions could intercept sensitive payment data before encryption, especially during the card details submission flow.	- Implement browser integrity checks - Use secure iframe isolation for payment forms - Deploy client-side encryption - Educate users about secure browsing practices
	Client-Side Resource Exhaustion	Denial of Service	Medium	Open		The Customer Client process in the Customer/Internet zone could be targeted with resource exhaustion attacks through malicious scripts or excessive API calls, preventing legitimate customers from completing payments.	- Implement client-side rate limiting - Use resource quotas for browser operations - Deploy performance monitoring - Implement graceful degradation

(1) Customer logs into the merchant site (Data Flow)

Description: OAuth

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	OAuth Token Hijacking	Spoofing	High	Open		Flow '(1) Customer logs into the merchant site' uses OAuth over HTTPS crossing from Customer actor to Customer Client. OAuth tokens could be intercepted or hijacked through redirect URI manipulation or authorization code interception attacks.	- Implement PKCE for OAuth flows - Validate redirect URIs strictly - Use state parameters to prevent CSRF - Implement token binding

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Session Fixation Attack	Elevation of Privilege	Medium	Open		The login flow from Customer to Customer Client could be vulnerable to session fixation where an attacker sets a known session ID before authentication, gaining unauthorized access after the user logs in.	- Regenerate session IDs after successful authentication - Implement secure session management - Use secure, httpOnly, sameSite cookies - Monitor for suspicious session patterns

(2) Customer proceeds to payment page to make a purchase (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Page Redirect Manipulation	Tampering	High	Open		Flow '(2) Customer proceeds to payment page to make a purchase' over HTTPS from Customer to Customer Client could be intercepted to redirect users to phishing payment pages that steal credentials and payment information.	- Implement strict redirect validation - Use Content Security Policy - Deploy HSTS headers - Monitor for unauthorized redirects

(7) Customer provides card details and finalizes payment (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Credit Card Data Interception	Information Disclosure	High	Open		Flow '(7) Customer provides card details and finalizes payment' from Customer to Customer Client transmits sensitive payment card data. Even though it's within the same trust boundary, the data could be exposed through browser vulnerabilities or malware.	- Implement end-to-end encryption for card data - Use tokenization immediately upon entry - Deploy PCI DSS compliant card handling - Implement field-level encryption
	Payment Data Tampering	Tampering	High	Open		The payment finalization flow from Customer to Customer Client could be tampered with to modify transaction amounts, merchant details, or payment destinations before submission.	- Implement transaction signing - Use server-side validation of all payment parameters - Deploy integrity checks - Implement transaction confirmation mechanisms

(3) Customer Client sends order intent including order amount (6) Return PaymentIntent to the Customer Client (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Intent Manipulation	Tampering	High	Open		Bidirectional flow '(3) Customer Client sends order intent/(6) Return PaymentIntent' crosses trust boundaries between Customer/Internet and Merchant/Web zones. Order amounts or payment intents could be tampered with during transmission despite being over encrypted channel.	- Implement request signing with HMAC - Validate all payment parameters server-side - Use idempotency keys - Implement strict input validation

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Intent Replay Attack	Repudiation	Medium	Open		The bidirectional payment intent flow between Customer Client and Merchant Web Server crossing trust boundaries could be subject to replay attacks where previous valid requests are resent to duplicate transactions.	- Implement nonce-based request validation - Use timestamp validation - Deploy idempotency keys - Maintain transaction logs with audit trails

(9) Attempt payment (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Processing Manipulation	Tampering	High	Open		Flow '(9) Attempt payment' from Stripe API service to Stripe Payment Service within the Stripe/Web boundary could be intercepted internally to modify payment amounts or destinations.	- Implement service-to-service authentication - Use encrypted internal communications - Deploy integrity checks - Implement transaction verification

(10) Payment Response (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Response Forgery	Tampering	High	Open		Flow '(10) Payment Response' from Stripe Payment Service to Stripe API service could be forged to indicate successful payment when it actually failed, leading to goods/services delivery without payment.	- Implement response signing - Use cryptographic verification - Maintain transaction state validation - Implement two-phase commit protocols

(11) Return the PaymentIntent with status (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Status Manipulation	Tampering	High	Open		Flow '(11) Return the PaymentIntent with status' crosses from Stripe/Web zone to Customer/Internet zone. Payment status could be manipulated to show success when payment failed or vice versa.	- Implement end-to-end status verification - Use webhook confirmations - Deploy cryptographic signatures - Implement server-side status validation
	Payment Status Information Leakage	Information Disclosure	Medium	Open		The payment status flow from Stripe API to Customer Client crossing multiple trust boundaries could leak sensitive transaction details, payment methods, or customer information.	- Minimize data in responses - Implement field-level encryption - Use secure transport with TLS 1.3 - Mask sensitive payment details

(8) Customer Client send Stripe e.ConfirmCardPayment() (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Confirmation Spoofing	Spoofing	High	Open		Flow '(8) Customer Client send Stripe e.ConfirmCardPayment()' crosses from Customer/Internet zone to Stripe/Web zone. Attackers could spoof payment confirmations or inject malicious confirmation requests.	- Implement Stripe's client-side authentication - Use publishable API keys correctly - Validate payment methods server-side - Implement 3D Secure authentication
	API Key Exposure	Information Disclosure	High	Open		The direct flow from Customer Client to Stripe API service for payment confirmation may expose API keys or tokens in client-side code or network traffic.	- Use only publishable keys client-side - Implement key rotation policies - Monitor for key exposure - Use environment-specific keys

(5) Return PaymentIntent to the Merchant (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	PaymentIntent Response Tampering	Tampering	High	Open		Flow '(5) Return PaymentIntent to the Merchant' from Stripe API service to Merchant Web Server crosses trust boundaries. The payment intent response could be modified to alter payment parameters or status.	- Verify response signatures from Stripe - Implement webhook validation - Use secure channels with certificate pinning - Validate response integrity

(4) Merchant sends order information inc amount and currency (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Order Information Disclosure	Information Disclosure	Medium	Open		Flow '(4) Merchant sends order information inc amount and currency' crosses from Merchant/Web zone to Stripe/Web zone. Order details and amounts could be exposed if the connection is compromised or through logging mechanisms.	- Enforce TLS 1.3 for all API communications - Implement mutual TLS authentication - Minimize data in transit - Encrypt sensitive fields additionally

Merchant Web Server (Process)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Merchant Server Compromise	Elevation of Privilege	High	Open		The Merchant Web Server process in the Merchant/Web trust zone handles payment processing and communicates with both customer and Stripe boundaries. Compromise could lead to unauthorized access to payment systems and customer data.	- Implement least privilege access controls - Deploy runtime application self-protection (RASP) - Use secure coding practices - Regular security audits and penetration testing
	Merchant API Denial of Service	Denial of Service	Medium	Open		The Merchant Web Server accepts requests from the Customer Client across trust boundaries and could be overwhelmed with excessive API calls, preventing legitimate payment processing.	- Implement rate limiting per customer - Deploy DDoS protection - Use API gateways with throttling - Implement circuit breakers

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Transaction Log Tampering	Repudiation	Medium	Open		The Merchant Web Server processes payment transactions but may lack proper audit logging, allowing disputes about transaction occurrence or modification of transaction records.	- Implement immutable audit logs - Use cryptographic signing of log entries - Deploy centralized log management - Maintain transaction receipts with digital signatures

Stripe API service (Process)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Stripe API Impersonation	Spoofing	High	Open		The Stripe API service in the Stripe/Web zone handles payment processing from multiple sources. DNS hijacking or certificate spoofing could redirect API calls to malicious endpoints.	- Implement certificate pinning - Use DNSSEC validation - Monitor for certificate changes - Implement backup API endpoint validation
	API Rate Limit Bypass	Denial of Service	Medium	Open		The Stripe API service receives requests from both Customer Client and Merchant Web Server across trust boundaries. Coordinated attacks could overwhelm the API despite rate limiting.	- Implement distributed rate limiting - Use API keys with appropriate limits - Deploy circuit breakers - Monitor for abnormal API usage patterns

Stripe Payment Service (Process)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Payment Service Authorization Bypass	Elevation of Privilege	High	Open		The Stripe Payment Service in the Stripe/Web zone processes actual payments. Vulnerabilities could allow unauthorized payment processing or access to payment infrastructure.	- Implement strict authorization checks - Use principle of least privilege - Deploy zero-trust architecture - Regular security assessments
	Payment Processing Failure	Denial of Service	Medium	Open		The Stripe Payment Service could be targeted with resource exhaustion or malformed requests causing payment processing failures and business disruption.	- Implement redundant payment processing - Deploy auto-scaling infrastructure - Use circuit breakers and fallback mechanisms - Monitor service health continuously