BUSINESS LOGIC

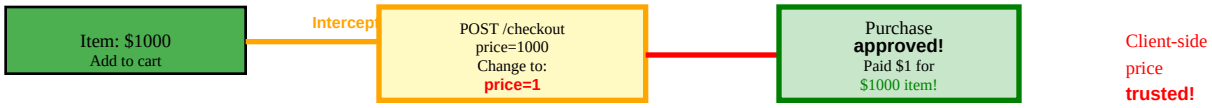
VULNERABILITIES

Complete Attack Reference

> **What are Business Logic Vulnerabilities?**
> Business logic flaws exploit intended functionality in unintended ways. Unlike technical vulnerabilities (XSS, SQLi), these abuse legitimate features through creative manipulation. They exploit assumptions developers make about user behavior. Often application-specific, requiring understanding of business workflows. Can't be detected by automated scanners.

1. Price Manipulation

Description: Modify price parameters to pay less or nothing. Intercept checkout requests, change price/quantity to negative values. Tamper with discount codes. Add items then change price to $0.01. No server-side price validation allows arbitrary pricing.



Example Attack:

```
POST /checkout price=1000 → Change to price=1 or price=-100 (credit!)
```

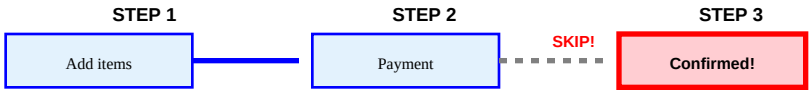
Attack Techniques:

```
Negative prices | Zero prices | Extreme discounts | Currency manipulation
```

> **Impact:** Financial loss, free products, revenue manipulation

2. Insufficient Workflow Validation

Description: Skip mandatory steps in multi-step processes. Complete purchase without payment. Access restricted content without subscription. Skip email verification, identity checks, approval workflows. Server doesn't validate step completion order.



Direct access to /order/confirm bypasses payment!

Example Attack:

```
Step 1: Add to cart → Step 2: Checkout → Skip to Step 4: Order confirmed!
```

Attack Techniques:

```
Step skipping | Direct endpoint access | Missing state validation
```

> **Impact:** Bypass payment, skip verification, unauthorized access

3. Race Conditions

Description: Exploit timing window in concurrent operations. Withdraw same money simultaneously from multiple ATMs. Use discount code multiple times. Redeem gift card in parallel requests. No atomic transaction handling allows double-spending.

Example Attack:

```
Simultaneous withdraw requests spend $100 balance twice = $200 withdrawn
```

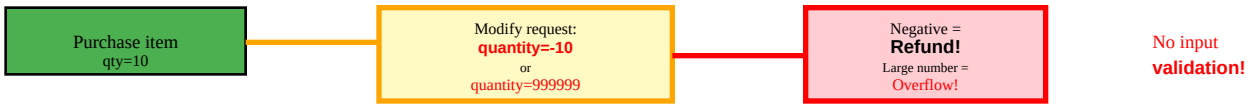
Attack Techniques:

```
Parallel requests | Multi-threaded exploitation | Timing attacks
```

Impact: Double-spending, balance manipulation, resource exhaustion

4. Parameter Tampering (Quantity/Amount)

Description: Modify quantity, amounts, IDs in requests. Negative quantities for refunds. Quantity overflow (2^31). Change user_id to someone else's. Tamper with loyalty points, credits, discounts applied.



Example Attack:

```
quantity=-10 generates refund instead of purchase | quantity=999999999 overflows
```

Attack Techniques:

```
Negative values | Integer overflow | Parameter substitution
```

Impact: Financial fraud, inventory manipulation, unauthorized access

5. Coupon/Voucher Abuse

Description: Reuse single-use codes. Stack incompatible discounts. Use expired coupons. Brute force coupon codes. Apply referral codes to own account. No proper validation allows unlimited discounts.

Example Attack:

```
Apply WELCOME10 code 10 times | Use code after expiration | Self-referral bonus
```

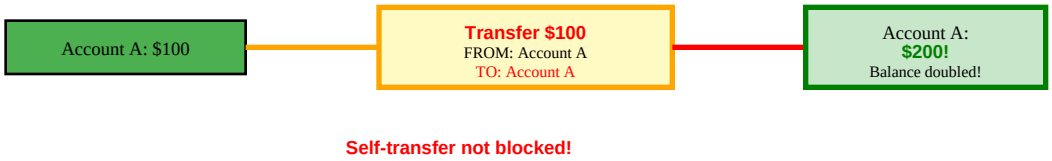
Attack Techniques:

```
Code reuse | Stacking | Brute forcing | Self-referral | Expired code use
```

> **Impact:** Revenue loss, unlimited discounts, referral fraud

6. Account/Balance Manipulation

Description: Manipulate credits, points, currency balances. Transfer from account A to A (double balance). Exploit rounding errors. Currency arbitrage between rates. Negative balance exploitation.



Example Attack:

```
Transfer $100 from account to itself → Balance increases to $200
```

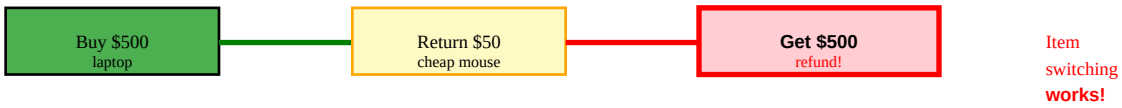
Attack Techniques:

```
Self-transfer | Rounding exploits | Currency conversion abuse
```

> **Impact:** Unlimited credits, financial fraud, economy breaking

7. Refund/Return Abuse

Description: Request refund without returning item. Return cheaper item than purchased. Multiple refunds for same order. Abuse no-questions-asked policies. Exploit automated refund systems.



Example Attack:

```
Buy expensive item, return cheap one, keep expensive | Multiple refund requests
```

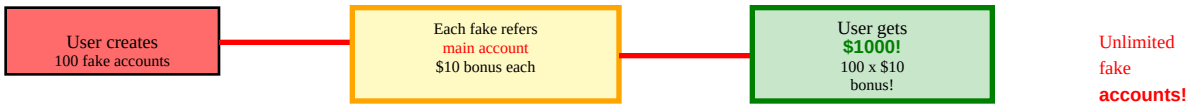
Attack Techniques:

```
Item switching | Multiple refund requests | Policy exploitation
```

Impact: Financial loss, inventory fraud, policy abuse

8. Referral/Reward System Gaming

Description: Create fake accounts for referral bonuses. Refer yourself with multiple accounts. Circular referral chains. Bot armies for rewards. Exploit unlimited referral bonuses.



Example Attack:

```
User creates 100 fake accounts, refers them all, collects 100x signup bonus
```

Attack Techniques:

```
Fake account creation | Self-referral | Automated bots | Circular chains
```

Impact: Bonus abuse, financial loss, metric manipulation

9. Rate Limiting Bypass (Business Impact)

Description: Bypass rate limits for business-critical actions. Mass password reset emails. Unlimited prize draws/lottery entries. Exhaust limited inventory instantly. OTP flooding. Resource reservation abuse.

Example Attack:

```
Enter lottery 10000 times, win guaranteed | Reserve all restaurant slots
```

Attack Techniques:

```
Request flooding | Distributed requests | IP rotation | Session manipulation
```

Impact: Service disruption, unfair advantage, resource exhaustion

10. Time Manipulation

Description: Manipulate timestamps, timezones, expiration times. Access time-limited content outside window. Exploit timezone conversion. Extend trial periods. Time travel attacks on subscriptions.

Example Attack:

```
Set system time to year 2099, access lifetime subscription | Timezone exploitation
```

Attack Techniques:

```
Client-side time manipulation | Timezone abuse | Expiry extension
```

Impact: Bypass time restrictions, extended trials, free access

11. Inventory/Stock Manipulation

Description: Purchase more than available stock. Reserve all inventory preventing others. Negative stock purchases. Hold items in cart indefinitely. Race condition on last item.

Example Attack:

```
Only 1 item left, 10 users purchase simultaneously = overselling
```

Attack Techniques:

```
Overselling | Stock hoarding | Cart manipulation | Race conditions
```

> **Impact:** Unfillable orders, stock manipulation, competitive advantage

12. Subscription/Billing Logic Flaws

Description: Cancel subscription but retain access. Downgrade without losing premium features. Free trial extensions. Payment failure doesn't revoke access. Subscription stacking for discount.

Example Attack:

```
Cancel subscription, features still work | Start 100 free trials same email
```

Attack Techniques:

```
Cancel without revocation | Trial abuse | Downgrade exploitation
```

> **Impact:** Free premium access, billing bypass, revenue loss

COMMON BUSINESS LOGIC SCENARIOS

E-commerce Specific

- Apply unlimited discount codes to single purchase
- Buy premium item at economy shipping price
- Use gift card balance multiple times in parallel
- Purchase with negative quantity for refund
- Access premium product pages, add to cart, modify price
- Exploit early-bird discounts after period ends
- Stack employee, student, senior discounts together
- Return different item than purchased

Financial/Banking

- Transfer money from account to itself (balance doubling)
- Withdraw more than balance via race condition
- Currency conversion rate manipulation
- Rounding error exploitation in transactions
- Overdraft without penalty
- Interest calculation manipulation

Gaming/Gambling

- Unlimited lottery entries
- Bet on both outcomes then cancel losing bet
- Exploit game RNG prediction
- Purchase in-game currency during processing lag
- Duplicate virtual items via race condition
- Negative gem/coin purchases

Subscription Services

- Unlimited free trial via new emails/cards
- Cancel subscription, retain premium access
- Downgrade plan, keep premium features
- Family plan abuse (unlimited accounts)
- Student discount without verification
- Pause subscription indefinitely

Booking/Reservation

- Book same slot multiple times
- Reserve without payment, hold inventory
- Modify booking after confirmation (upgrade)
- Cancel and rebook to exploit price changes
- Overbooking via concurrent requests

Authentication/Access

- Access premium features in trial mode
- Skip email verification, access account
- Bypass 2FA during password reset
- Access admin panel without role check
- Share single account credentials unlimited times

TESTING METHODOLOGY

1. Understand Business Flows: Map all critical business processes: checkout, payment, registration, refunds, subscriptions. Document expected workflow steps. Identify valuable actions: purchasing, credits, rewards. Understanding is key.

2. Identify Assumptions: What does application assume? That users follow steps in order? That prices come from server? That users won't send parallel requests? List all implicit assumptions developers made.

3. Think Like User (Good and Bad): Legitimate use: How should it work? Malicious use: How can I abuse this for profit/access? What would dishonest user try? Consider attacker motivation.

4. Test Parameter Manipulation: Try: negative values, zero values, extremely large numbers, null values, strings instead of numbers. Modify prices, quantities, IDs, amounts. Use Burp Suite to intercept and modify.

5. Test Workflow Violations: Skip steps in multi-step processes. Go backwards in workflow. Access later steps directly via URL. Submit incomplete forms. Test out-of-order operations.

6. Test Race Conditions: Use Burp Turbo Intruder or custom scripts. Send simultaneous requests for: purchases, withdrawals, code redemptions, reservations. Look for double-processing, inconsistent state.

7. Test Boundary Conditions: What happens at limits? Buy 0 items, negative items, maximum integer. Test minimum balances, maximum discounts. Edge cases reveal logic flaws.

8. Test State Manipulation: Can you reuse tokens/codes? Access features after downgrade? Use expired coupons? Maintain access after cancellation? State management critical.

9. Test Timing Issues: Manipulate client-side timestamps. Exploit timezone differences. Access time-limited content outside windows. Extend trials by time manipulation.

10. Think Creative & Unusual: What would attacker profit from? How to get free stuff? Unlimited credits? Business logic requires creativity. Try unexpected combinations. Think outside normal usage.

DETECTION CHECKLIST

✓ Map all critical business workflows completely

✓ Document expected step-by-step process flow

✓ Identify all financial transactions and calculations

✓ Test negative values in price, quantity, amount fields

✓ Try zero values (price=0, quantity=0)

✓ Test extremely large numbers (integer overflow)

✓ Skip steps in multi-step processes

✓ Access later workflow steps directly via URL

✓ Test parallel/simultaneous requests on critical actions

✓ Try reusing single-use codes/tokens/vouchers

✓ Test self-referral in referral systems

✓ Apply multiple discount codes simultaneously

✓ Test expired coupon codes

✓ Attempt balance manipulation (self-transfer)

✓ Test currency conversion at different rates

✓ Try multiple refund requests for same order

✓ Reserve limited resources (slots, inventory)

✓ Test time manipulation (client-side timestamps)

✓ Cancel subscription, check if access persists

✓ Start multiple free trials with same details

✓ Test downgrade, check for premium feature retention

✓ Modify parameters in payment confirmation requests

✓ Test rate limits on business-critical actions

✓ Look for client-side price calculations

✓ Check for proper atomic transactions

PREVENTION & MITIGATION

• **Server-Side Validation Always:** Never trust client input. Validate all parameters server-side: prices, quantities, IDs, steps completed. Recalculate totals server-side. Don't accept prices from client.

• **Implement Proper State Management:** Track workflow state server-side. Verify prerequisites before allowing action. Use session variables to enforce step order. Prevent direct access to later steps.

• **Use Atomic Transactions:** Database transactions must be atomic. Use SELECT FOR UPDATE for balance checks. Implement proper locking mechanisms. Prevent race conditions in critical operations.

• **Implement Rate Limiting:** Limit business-critical actions per user/IP/session. Prevent mass code redemption, unlimited entries, resource hoarding. Use sliding windows, not fixed time periods.

• **Idempotency Keys:** Generate unique keys for financial transactions. Prevent duplicate processing of same request. Check idempotency before processing. Critical for payments, withdrawals.

• **Voucher/Code Management:** Single-use codes must be marked as used atomically. Check expiration server-side. Validate code against user/order. Implement redemption limits. Track usage history.

• **Balance Checks:** Always verify sufficient balance before deduction. Check balance at transaction time, not earlier. Atomic balance read and update. Prevent negative balances unless intended.

• **Logging and Monitoring:** Log all financial transactions completely. Alert on anomalies: negative purchases, extreme discounts, unusual patterns. Review logs for abuse patterns. Implement fraud detection.

• **Business Rules in Code:** Explicit validation of business rules. Document assumptions. Code reviews for logic flaws. Security team understands business context. Threat model business processes.

• **Time-Based Validations:** Server-side time checks always. Don't trust client timestamps. Validate expiry, trial periods server-side. Use UTC consistently. Account for timezones properly.

BUSINESS LOGIC RED FLAGS

Red Flag	Risk	Example
Client-side price calc	Price manipulation	JavaScript calculates total
No workflow validation	Step skipping	Can access step 4 directly
Reusable tokens/codes	Code abuse	Same voucher works twice
No rate limits	Mass abuse	Unlimited lottery entries
Integer fields unchecked	Overflow/negative	quantity=-1000
Missing balance checks	Overdraft	Withdraw without funds
Time on client	Time manipulation	Trial expiry checked client-side
No concurrency control	Race conditions	Parallel withdrawals succeed
Static transaction IDs	Replay attacks	Resubmit payment
Trust on referrer	Referral gaming	Self-referral allowed

TESTING TOOLS

Tool	Purpose	Use Case
Burp Suite	Intercept and modify	Parameter tampering, workflow testing
Burp Intruder	Automated attacks	Brute force codes, test ranges
Burp Turbo Intruder	Race conditions	Parallel requests, timing attacks
Postman	API testing	Workflow manipulation, state testing
Python/Scripts	Custom automation	Complex race conditions, mass testing
Browser DevTools	Client-side inspection	Find hidden parameters, client logic

Note: This cheat sheet is for educational and authorized security testing only. Unauthorized exploitation of business logic for financial gain is fraud and illegal. Always obtain written permission before testing.