Detecting Logic Vulnerabilities in E-Commerce Applications

Fangqi Sun, Liang Xu, Zhendong Su

UC DAVIS

RBS WorldPay™

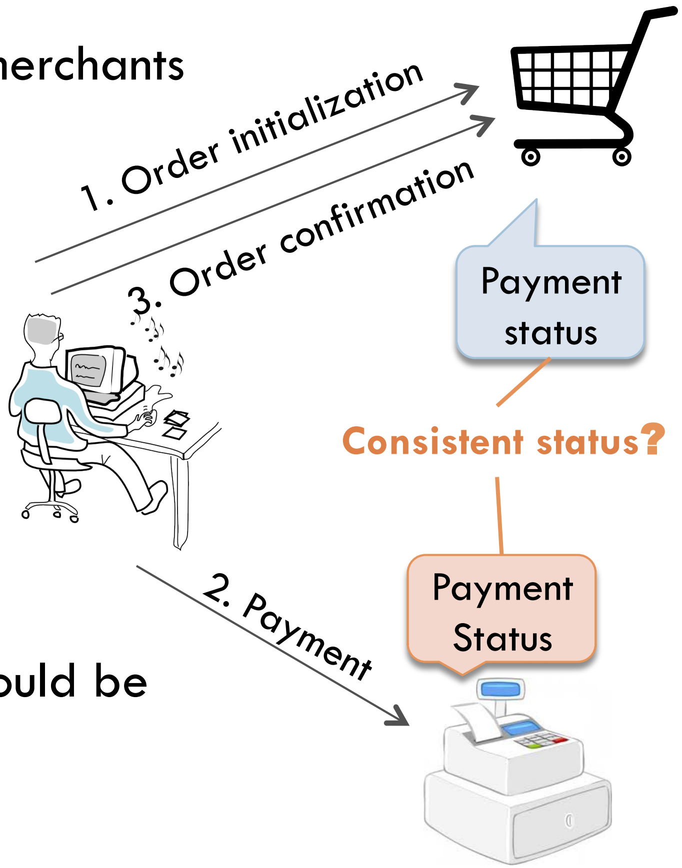
Authorize.Net®
a CyberSource solution

PayPal™

oscommerce

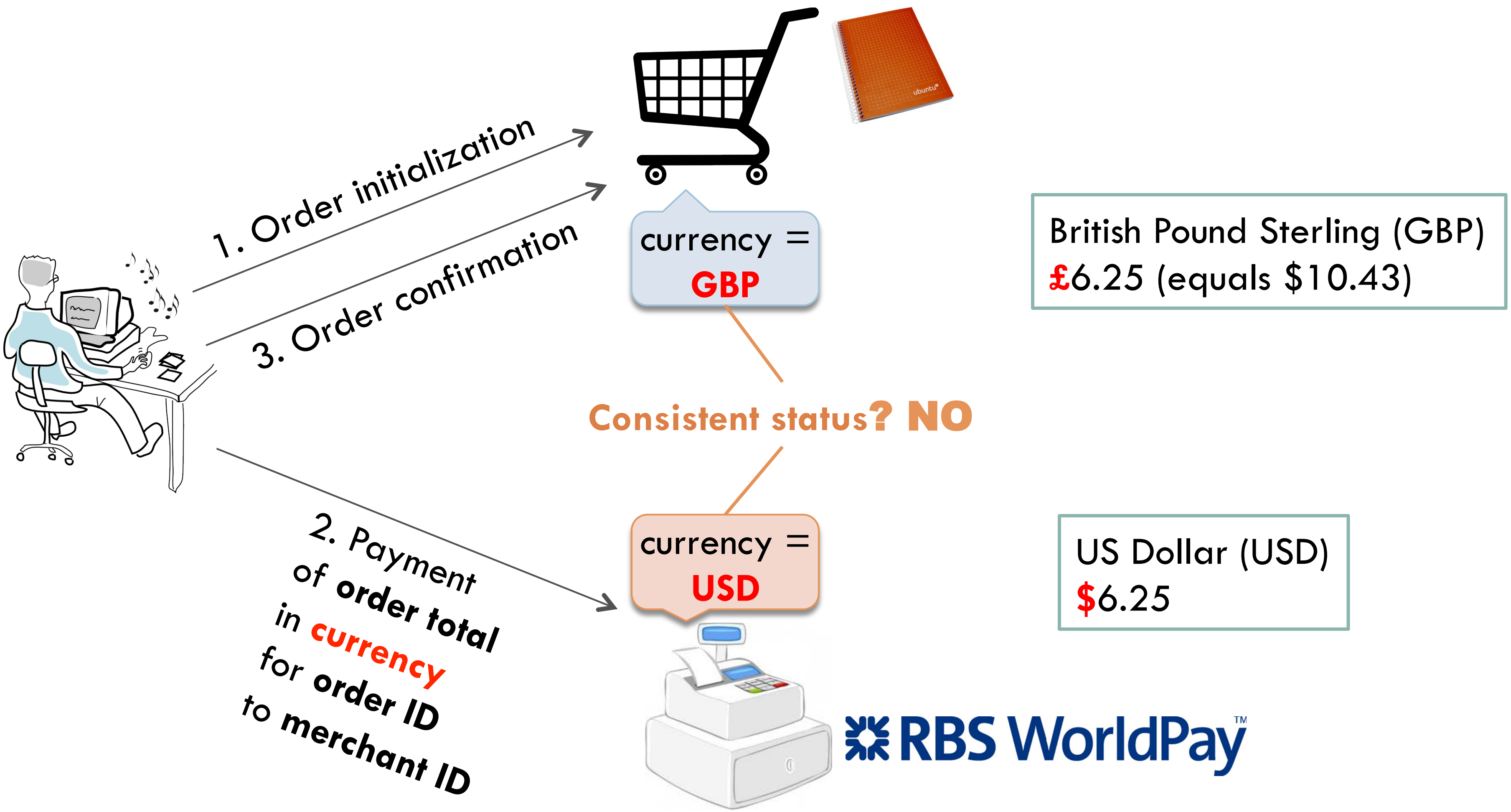
Logic Vulnerabilities in E-Commerce Web Applications

- Third-party cashiers
  - Bridge the trustiness gap between customers and merchants
  - Complicate logic flows during checkout

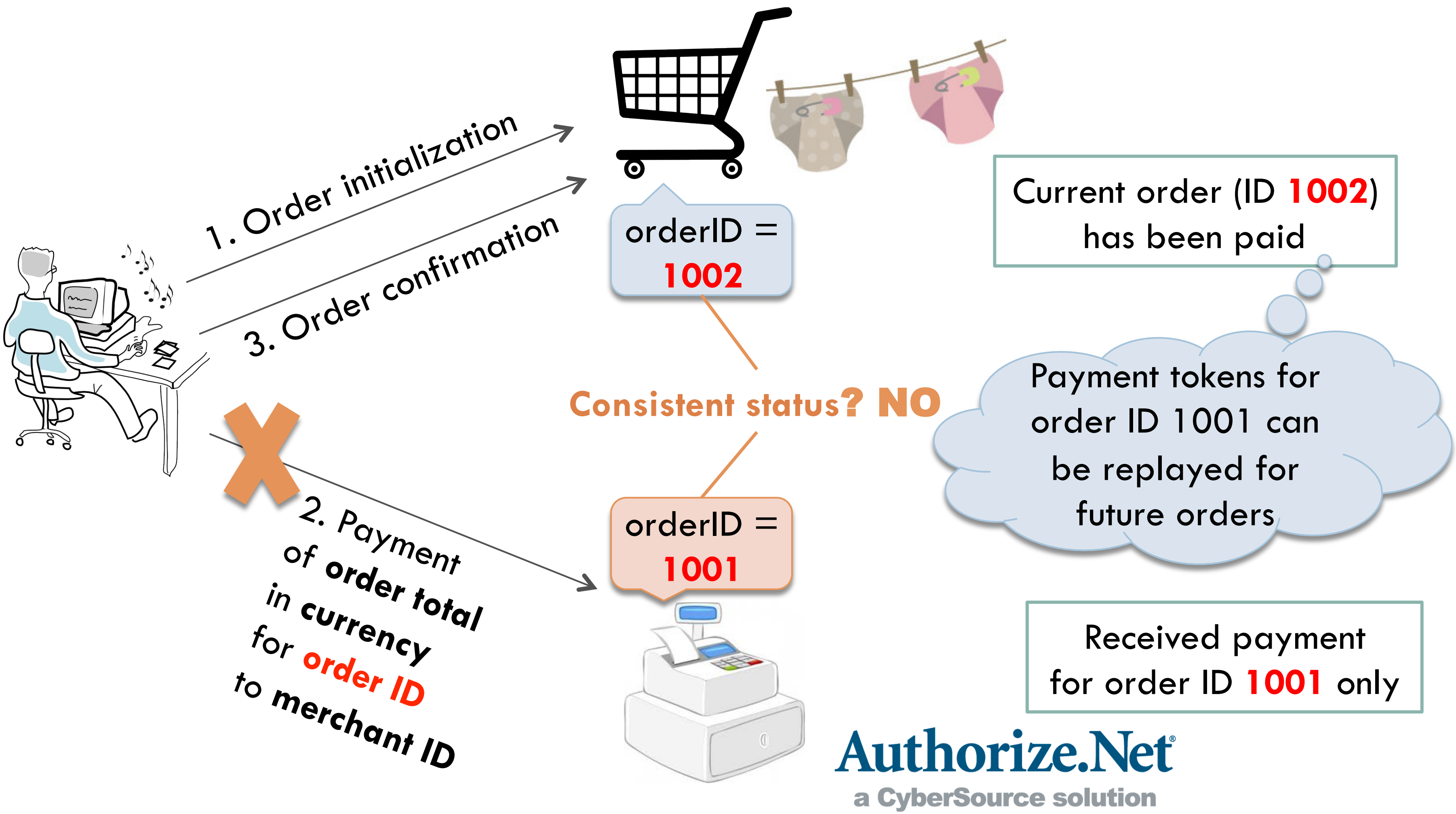


- Logic vulnerabilities in e-commerce web applications
  - Abuse application-specific functionality
  - Allow attackers to purchase products or services with incorrect or no payment
  - Have multiple attack vectors
    - Assumptions of **user inputs** and **user actions** should be explicitly checked
  - Example
    - CVE-2009-2039 is reported for Luottokunta (v1.2) but the patched Luottokunta (v1.3) is still vulnerable

Attack on Currency



Attack on Order ID



Attack on Merchant ID

Key Challenge

- Logic vulnerabilities in e-commerce web applications are application-specific
  - Thorough code review of all possible logic flows is non-trivial
  - Various application-specific logic flows, cashier APIs and security checks make automated detection difficult

- Key challenge of automated detection

The lack of a general and precise notion of **correct payment logic**

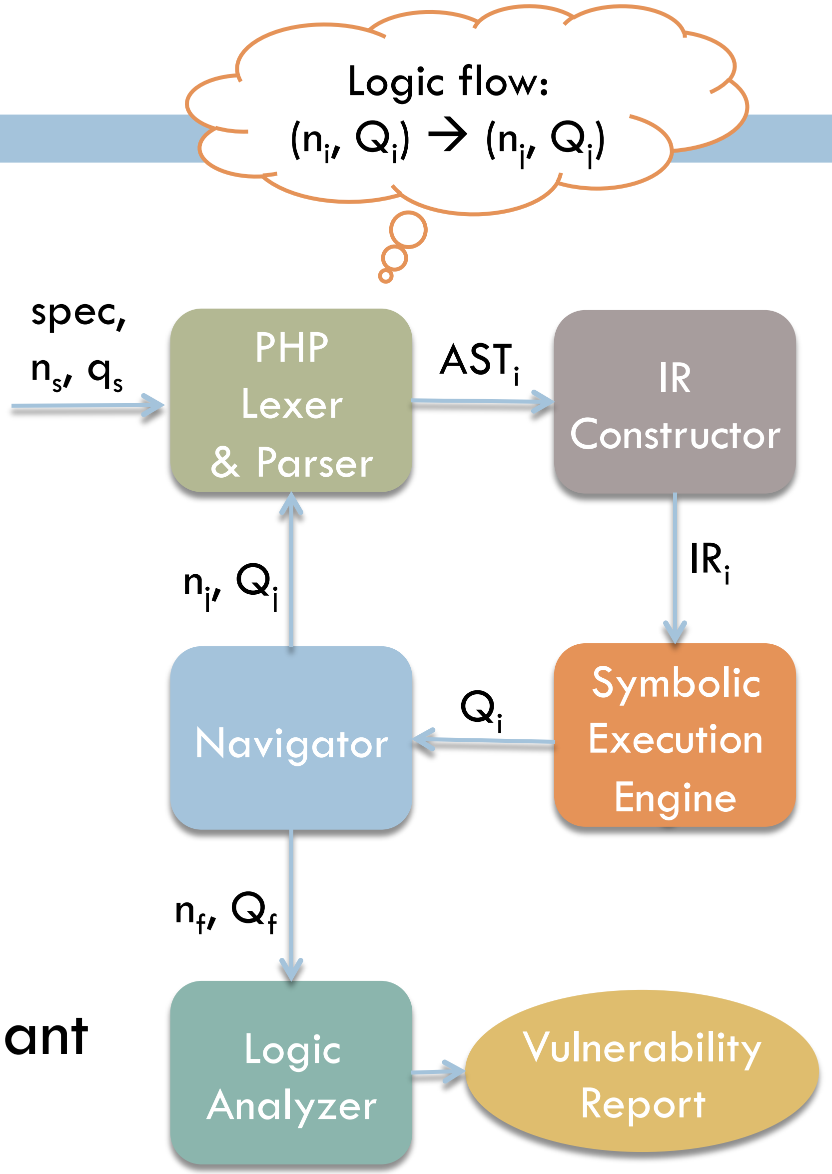
Key Insight

- A **common invariant** for automated detection

A checkout is secure when it guarantees the **integrity** and **authenticity** of critical payment status (order ID, order total, merchant ID and currency)

Our Approach

- A symbolic execution framework that explores critical control flows exhaustively
- Tracking taint annotations across checkout nodes
  - Payment status
  - Exposed signed token (signed with a cashier-merchant secret)



Taint Removal Rules

- Conditional checks of (in)equality
  - When an untrusted value is verified against a trusted one
  - Example of removing taint from order total

md5(SECRET . $_SESSION['order']→info['total']) == md5(SECRET . $_GET['oTotal'])

- Writes to merchant databases
  - When an untrusted value is included in an INSERT/UPDATE query
  - Merchant employee can easily spot tampered values
- Secure communication channels (merchant-to-cashier cURL requests)
  - Remove taint from order ID, order total, merchant ID or currency when such components are present in request parameters

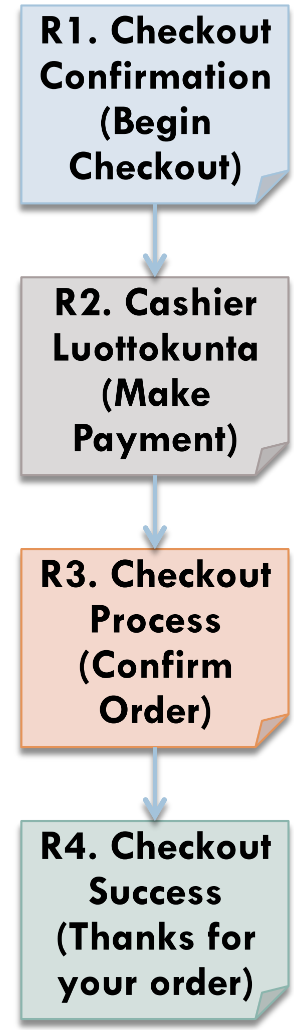
Taint Addition Rule

- Add an exposed signed token when used in a conditional check of a cashier-to-merchant request
  - Security by obscurity is insufficient

- Example
  - Hidden HTML form element: md5($secret . $orderId . $orderTotal)
  - $_GET['hash'] == md5($secret . $_GET['oId'] . $_GET['oTotal'])
  - This exposed signed token md5($secret . $orderId . $orderTotal) nullifies checks on order ID and order total

Vulnerability Detection Example

- R1. User → Merchant(checkoutConfirmation.php)
  - Symbolic HTML form contains two URLs: cashier URL and return URL(checkoutProcess.php).
- R2. User → Cashier(https://dmp2.luottokunta.fi)
  - Modeling cashier as trusted black box
- R3. User → Merchant(checkoutProcess.php), redirection
  - Representing all possible cashier responses with symbolic inputs
- R4. User → Merchant(checkoutSuccess.php), redirection
  - Analyzing logic states at this destination node (end of checkout) to detect logic vulnerabilities



Luottokunta (v1.3)

```
1.  function before_process() {
2.    if (!isset($_GET['orderID'])) {
3.      tep_redirect(FILE_PAYMENT);
4.    } else {
5.      $orderID = $_GET['orderID'];
6.    }

7.    $price = $_SESSION['order']-
    >info['total'];
8.    $tarkiste = SECRET_KEY . $price
9.                      . $orderID .
    MERCHANT_ID;
10.   $mac = strtoupper(md5($tarkiste));

11.   if (($_POST['LKMAC'] != $mac)
12.        && ($_GET['LKMAC'] != $mac)) {
13.     tep_redirect(FILE_PAYMENT);
14.   } else {
15.     ...
16.}
17.}
```

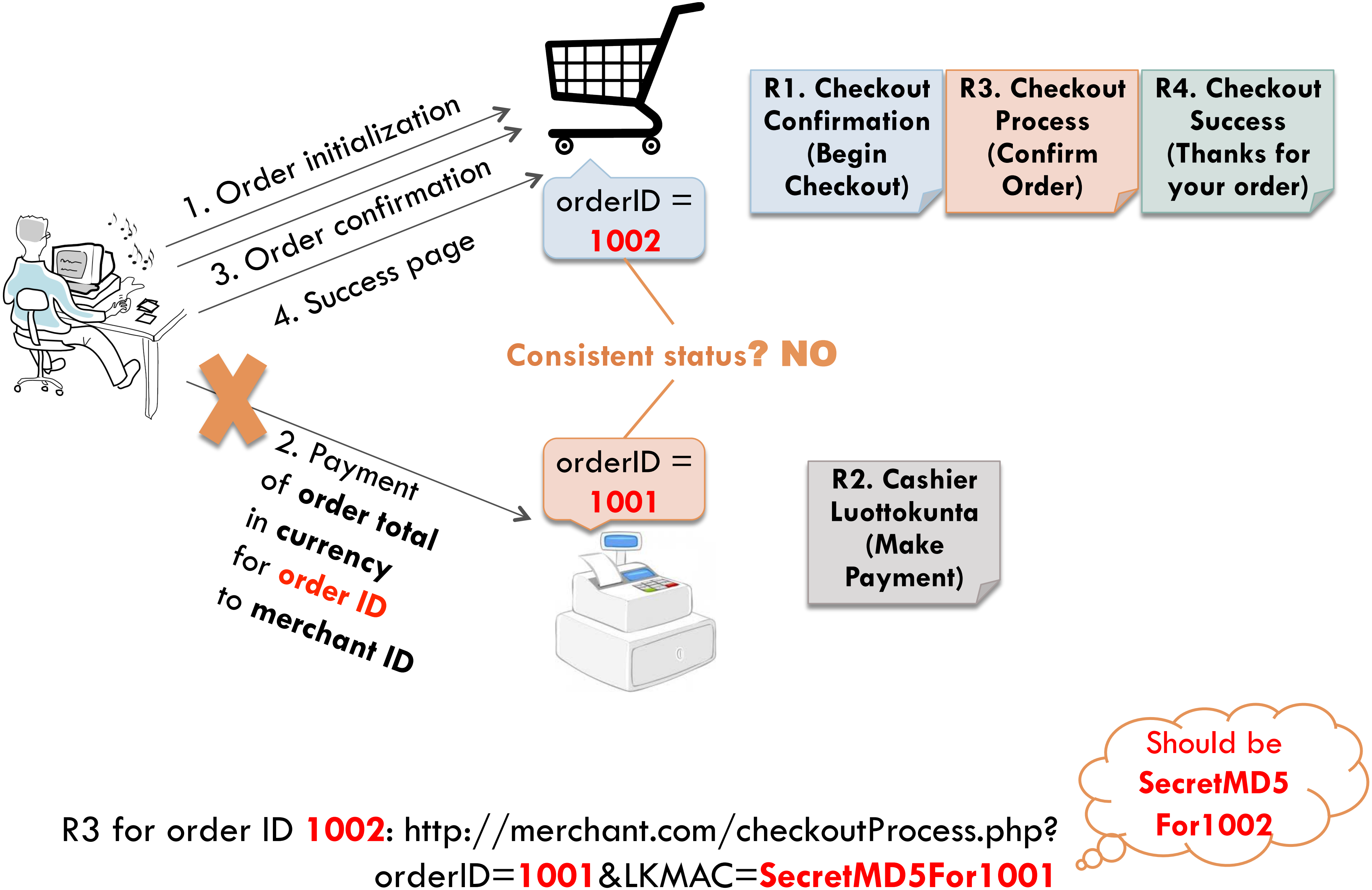
R3. Checkout Process (Confirm Order)

Path condition for 'else branch' (line 15):

```
[ or
  ($_POST['LKMAC'] =
      strtoupper(md5(SECRET_KEY
    . $_SESSION['order']->info['total']
    . $_GET['orderID'] . MERCHANT_ID)));

  ($_GET['LKMAC'] = ...);
]
```

- Remove taint from **order total** ($_SESSION['order']->info['total']) and **merchant ID** (MERCHANT_ID).
- **Order ID** and **currency** are still tainted: $_GET['orderID'] is an untrusted user input.
- 'If' branch is a backward logic flow; 'else' branch is a forward logic flow



1. Order initialization
3. Order confirmation
4. Success page
2. Payment of order total in currency for order ID to merchant ID
orderID = 1002
orderID = 1001
Consistent status? NO
R1. Checkout Confirmation (Begin Checkout)
R3. Checkout Process (Confirm Order)
R4. Checkout Success (Thanks for your order)
R2. Cashier Luottokunta (Make Payment)
Should be SecretMD5 For1002
R3 for order ID 1002: http://merchant.com/checkoutProcess.php?orderID=1001&LKMAC=SecretMD5For1001

Evaluation

oscommerce

- Subjects: 22 unique payment modules of osCommerce
  - More than 14,000 registered websites, 928 payment modules, 13 years of history (osCommerce v2.3)
  - **20** out of 46 default modules with distinct CFGs
  - **2** Luottokunta payment modules (v1.2 & v1.3)
- Metrics
  - Effectiveness: Detected 12 logic vulnerabilities (11 new) with no false positives
  - Performance

Logic Vulnerability Analysis Results

Payment Module	Safe	Payment Module	Safe
2Checkout	✗	PayPal Pro - Direct Payments	✔
Authorize.net CC AIM	✔	PayPal (Payflow) - Direct Payments	✔
Authorize.net CC SIM	✗	PayPal (Payflow) - Express Checkout	✔
ChronoPay	✗	PayPal Standard	✗
inpay	✔	PayPoint.net SECPay	✗
iPayment (Credit Card)	✗	PSiGate	✗
Luottokunta (v1.2)	✗	RBS WorldPay Hosted	✗
Luottokunta (v1.3)	✗	Sage Pay Direct	✔
Moneybookers	✓	Sage Pay Form	✗
NOCHEX	✗	Sage Pay Server	✔
PayPal Express	✔	Sofortüberweisung Direkt	✔*

Taint Annotations of 12 Vulnerable Payment Modules

Payment Module	Order Id	Order Total	Merchant Id	Currency	Signed Tokens
2Checkout	✗	✗	✗	✗	
Authorize.net SIM	✗			✗	
ChronoPay	✗	✗	✗	✗	✗
iPayment (Credit card)	✗				
Luottokunta (v1.2)	✗	✗	✗	✗	
Luottokunta (v1.3)	✗			✗	
NOCHEX	✗	✗	✗	✗	
PayPal Standard			✗		
PayPoint.net SECPay	✗	✗		✗	
PSiGate	✗	✗	✗	✗	
RBS WorldPay Hosted				✗	✗
Sage Pay Form		✗		✗	
Total	**9**	**7**	**6**	**10**	**2**

Performance Results of 12 Vulnerable Payment Modules

Payment Module	Files	Nodes	Edges	Stmts	States	Flows	Time(s)
2Checkout	105	5,194	6,176	8,385	40	4	16.04
Authorize.net SIM	105	5,221	6,221	8,435	46	4	16.89
ChronoPay	99	5,013	5,969	8,084	69	5	31.51
iPayment (Credit card)	99	4,999	5,932	7,918	38	5	21.86
Luottokunta (v1.2)	105	5,158	6,127	8,291	34	4	15.33
Luottokunta (v1.3)	105	5,164	6,135	8,308	35	4	15.33
NOCHEX	105	5,145	6,111	8,237	33	4	15.03
PayPal Standard	99	5,040	6,006	8,170	68	6	33.01
PayPoint.net SECPay	105	5,174	6,152	8,332	40	4	15.80
PSiGate	106	5,231	6,228	8,436	44	4	16.82
RBS WorldPay Hosted	99	5,019	5,977	8,121	79	5	36.12
Sage Pay Form	106	5,315	6,329	8,762	55	4	19.96
Average of 22	**102.73**	**5,173**	**6,162**	**8,376**	**67.27**	**5.05**	**31.43**

Conclusion

- First static detection of logic vulnerabilities in e-commerce applications
  - Based on an application-independent invariant
  - A scalable symbolic execution framework for PHP applications, incorporating taint tracking of payment status
- Three responsible proof-of-concept experiments on live websites
- Evaluated our tool on 22 unique payment modules and detected 12 logic vulnerabilities (11 are new)