Detecting Logic Vulnerabilities in E-Commerce Applications

Fangqi Sun, Liang Xu, Zhendong Su
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

1. Order initialization
2. Payment of order total in currency for order ID to merchant ID
3. Order confirmation

Consistent status? NO

currency = GBP

British Pound Sterling (GBP) £6.25 (equals $10.43)

currency = USD

US Dollar (USD) $6.25
Attack on Order ID

1. Order initialization
2. Payment of order total in currency for order ID to merchant ID
3. Order confirmation

Current order (ID 1002) has been paid

Consistent status? NO

Payment tokens for order ID 1001 can be replayed for future orders

Received payment for order ID 1001 only
Payee is chocolateDelight

merchantID = chocolateDelight

Consistent status? NO

Payee is attackerAlice

merchantID = attackerAlice

Setting up a PayPal merchant account for Alice is easy
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['old'] . $_GET['oTotal'])`
  - This exposed signed token `md5($secret . $orderId . $orderTotal)` nullifies checks on order ID and order total
Vulnerability Detection Example

  - Symbolic HTML form contains two URLs: cashier URL and return URL(checkoutProcess.php).

  - 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 . $orderId . MERCHANT_ID;
9.     $mac = strtoupper(md5($tarkiste));
10.    if (($_POST['LKMAC'] != $mac) && ($_GET['LKMAC'] != $mac)) {
11.        tep_redirect(FILE_PAYMENT);
12.    } else {
13.        ...
14.    }
15.}

Path condition for ‘else branch’ (line 15):
[ or
($_POST['LKMAC'] =
 strtoupper(md5(SECRET_KEY . $_SESSION['order']->info['total'] . $_GET['orderID'] . MERCHANT_ID)));
($_GET['LKMAC'] = ...
]

R3. Checkout Process (Confirm Order)

- 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.
R3 for order ID 1002: http://merchant.com/checkoutProcess.php?
orderID=1001&LKMAC=SecretMD5For1001
Evaluation

- **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

<table>
<thead>
<tr>
<th>Payment Module</th>
<th>Safe</th>
<th>Payment Module</th>
<th>Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>2Checkout</td>
<td>✗</td>
<td>PayPal Pro - Direct Payments</td>
<td>✓</td>
</tr>
<tr>
<td>Authorize.net CC AIM</td>
<td>✓</td>
<td>PayPal (Payflow) - Direct Payments</td>
<td>✓</td>
</tr>
<tr>
<td>Authorize.net CC SIM</td>
<td>✗</td>
<td>PayPal (Payflow) - Express Checkout</td>
<td>✓</td>
</tr>
<tr>
<td>ChronoPay</td>
<td>✗</td>
<td>PayPal Standard</td>
<td>✗</td>
</tr>
<tr>
<td>inpay</td>
<td>✓</td>
<td>PayPoint.net SECPay</td>
<td>✗</td>
</tr>
<tr>
<td>iPayment (Credit Card)</td>
<td>✗</td>
<td>PSiGate</td>
<td>✗</td>
</tr>
<tr>
<td>Luottokunta (v1.2)</td>
<td>✗</td>
<td>RBS WorldPay Hosted</td>
<td>✗</td>
</tr>
<tr>
<td>Luottokunta (v1.3)</td>
<td>✗</td>
<td>Sage Pay Direct</td>
<td>✓</td>
</tr>
<tr>
<td>Moneybookers</td>
<td>✓</td>
<td>Sage Pay Form</td>
<td>✗</td>
</tr>
<tr>
<td>NOCHEX</td>
<td>✗</td>
<td>Sage Pay Server</td>
<td>✓</td>
</tr>
<tr>
<td>PayPal Express</td>
<td>✓</td>
<td>Sofortüberweisung Direkt</td>
<td>✓*</td>
</tr>
</tbody>
</table>
## Taint Annotations of 12 Vulnerable Payment Modules

<table>
<thead>
<tr>
<th>Payment Module</th>
<th>Order Id</th>
<th>Order Total</th>
<th>Merchant Id</th>
<th>Currency</th>
<th>Signed Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>2Checkout</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Authorize.net SIM</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>ChronoPay</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>iPayment (Credit card)</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Luottokunta (v1.2)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Luottokunta (v1.3)</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOCHEX</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>PayPal Standard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>PayPoint.net SECPay</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSiGate</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>RBS WorldPay Hosted</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Sage Pay Form</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>9</strong></td>
<td><strong>7</strong></td>
<td><strong>6</strong></td>
<td><strong>10</strong></td>
<td><strong>2</strong></td>
</tr>
</tbody>
</table>
## Performance Results of 12 Vulnerable Payment Modules

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