Detection of Logic Flaws in Web Applications

Davide Balzarotti

Giancarlo Pellegrino
Distributed, Service-oriented, Web-based applications

Order item $I$

Transfer $\text{value}(I)$ to $S$

Shipping in 2 days
Distributed, Service-oriented, Web-based applications

Order item $I$

Transfer $\text{value}(I)$ to $S$

Shipping in 2 days
Distributed, Service-oriented, Web-based applications

Order item $I$

Log $U$ in to $S$

Grant $U$ the access to $S$

Transfer $\text{value}(I)$ to $S$

Log $U$ in to $P$

Grant $U$ the access for paying $I$

Shipping in 2 days
Distributed, Service-oriented, Web-based applications

Order item I

Log U in to S

Grant U the access to S

Transfer value(I) to S

Log U in to P

Grant U the access for paying I

Shipping in 2 days
Security Risks

- User (U) logs in to IdP
- IdP authenticates U and signs the order item I
- AS verifies the signature and signs it
- Seller Inc. (S) verifies the signature and signs it
- Bank Inc. (P) verifies the signature and signs it
- Fraudulent U requests the access to S
- S grants the access to U
- S transfers the order to P
- P transfers the order to S
- S ships the item in 2 days
- No payment is transferred
Threats to Business Applications

- **Increased attack surface:**
  - monolithic and centralized → n-tier and distributed over the Web

- **In 2012:**
  - 92% of the attacks from external attackers
  - +33% of web-attacks from 2011
  - Web-attacks costed in avg >$1M per organization

- A large number of techniques exist to test web applications for the presence of several **classes** of vulnerabilities
  - SQL injection
  - Parameter pollution
  - ...
Logic Vulnerabilities

- Still lack a formal definition
  - Design flaws, business logic errors, …
  - “Weaknesses [...] that commonly allow attackers to manipulate the business logic of an application.” – CWE Database

- Mainly caused by insufficient validation of the workflow and/or data flow of the application
  - Detecting logic vulnerabilities requires a model of the application logic

- Logic Vulnerabilities can exhibit patterns, e.g.
  - Information disclosure
  - Authentication bypass
The Rise of Logic Vulnerabilities

- Poorly studied

- Increasing Importance:
  - Improper authentication overtook XSS in OWASP 2013 Top10
  - Logic flaws 2nd security risk in 2013 according to TrustWave

- Number of CVE entries associated to logic flaws over the years is increasing
### State of the Art

<table>
<thead>
<tr>
<th>Source code</th>
<th>Formal Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

- **White-box testing** [BalzarottiCCS07, FelmetsgerUSENIX10, ...]
  - Scalability issues
  - Not applicable for business applications for which source code is not available
State of the Art

- **White-box testing**

- **Design Verification via Model Checking**  
  - Attacks discovered by the model checker are only valid at model-level  
  - Attack interpretation and execution against implementations is done manually
### State of the Art

<table>
<thead>
<tr>
<th>Source code</th>
<th>Formal Model</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>White box</td>
<td>Black box</td>
<td>Design Verification</td>
</tr>
<tr>
<td>No</td>
<td>Black box</td>
<td>Black box</td>
<td>Design Verification</td>
</tr>
</tbody>
</table>

- **White-box testing**
- **Design Verification via Model Checking**

  - Mostly based on crawlers and fuzzers
  - Still unable to automatically detect logic flaws
  - The entire testing is mainly done manually
## State of the Art

<table>
<thead>
<tr>
<th>Source code</th>
<th>Formal Model</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Black box</td>
<td>White box</td>
<td>Black box</td>
</tr>
<tr>
<td></td>
<td>Design Verification</td>
<td>White box</td>
<td>Black box</td>
</tr>
<tr>
<td>No</td>
<td>Black box</td>
<td></td>
<td>Black box</td>
</tr>
<tr>
<td></td>
<td>Design Verification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Automated tools**

**Manual inspection**
Bridging the Gap

- From the **Model Checking** side
  - Requires abstract, simplified (but precise) models of the application/protocol
  - Flaws in the model needs to be manually verified in the real system

- From the **System Security Testing** side
  - Requires a real deployment of the application/protocol
  - Without a model, it is hard to detect logic flaws
Bridging the Gap

- From the **Model Checking** side
  - Requires abstract, simplified (but precise) models of the application/protocol
  - Flaws in the model needs to be manually verified in the real system

- From the **System Security Testing** side
  - Requires a real deployment of the application/protocol
  - Without a model, it is hard to detect logic flaws
Manual Testing

- **Understand** the web application
  - Intended workflow and data flow between pages
  - Model how the application *is supposed* to work

- **Design** tests that try unconventional behaviors
  - On the workflow plane (e.g., re-ordered steps)
  - On the data plane (e.g., replay tokens)

- **Run** tests

- **Observe** the results and **identify** vulnerabilities
Overview of the Approach

I - Model Inference

II - Behavioral Patterns Extraction

III – Test Cases Generation

IV – Test Cases Execution

[EURECOM]

[NDSS14]
Behavioral Patterns Extraction

I - Model Inference

II - Behavioral Patterns Extraction

III – Test Cases Generation

IV – Test Cases Execution
Workflow

Traces:

\[ \pi_1 = \langle a, b, a, c, d, e, f, e \rangle \]
\[ \pi_2 = \langle a, c, d, e, f, e \rangle \]

Navigation Graph:
Workflow

Traces:
\[\pi_1 = \langle a, b, a, c, d, e, f, e \rangle\]
\[\pi_2 = \langle a, c, d, e, f, e \rangle\]

Navigation Graph:

- **TrWP**: Trace Waypoints
- **St**: Singleton Nodes
- **MWP**: Multi-step Operations
- **Rp**: Repeatable Operations
- **MWP**: Model Waypoints

Graph edges and nodes are marked with these categories.
Data Flow

**Trace 1:**

   
   <HTML>
   <a href="/view.php?tid=23">[…]
   </a>
   
   
   <HTML>
   <a href="/add.php?tid=23">[…]
   </a>
   
   
   <HTML>
   <a href="/checkout">[…]
   </a>

**Trace 2:**


   <HTML>
   <a href="/add.php?tid=6">[…]
   </a>


   <HTML>
   <a href="/checkout">[…]
   </a>
Data Flow

Trace 1:

   
   `<HTML>
   <a href="/view.php?tid=23">[...]</a>
   </HTML>`

   
   `<HTML>
   <a href="/add.php?tid=23">[...]</a>
   </HTML>`

   
   `<HTML>
   <a href="/checkout">[...]</a>
   </HTML>`

Client Generated

Server Generated

24
Data Flow

**Trace 1:**

   
   <HTML>
   <a href="/view.php?tid=23">[…]</a>
   </HTML>

   
   <HTML>
   <a href="/add.php?tid=23">[…]</a>
   </HTML>

   
   <HTML>
   <a href="/checkout">[…]</a>
   </HTML>

**Trace 2:**

   
   <HTML>
   <a href="/add.php?tid=6">[…]</a>
   </HTML>

   
   <HTML>
   <a href="/checkout">[…]</a>
   </HTML>
Data Propagation Chains

<table>
<thead>
<tr>
<th>Trace $\pi_1$</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>Node</td>
</tr>
<tr>
<td>$p_{1,1}$</td>
<td>$a$</td>
</tr>
<tr>
<td>$p_{1,2}$</td>
<td>$b$</td>
</tr>
<tr>
<td>$p_{1,3}$</td>
<td>$c$</td>
</tr>
<tr>
<td>$p_{1,4}$</td>
<td>$b$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trace $\pi_2$</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>Node</td>
</tr>
<tr>
<td>$p_{2,1}$</td>
<td>$a$</td>
</tr>
<tr>
<td>$p_{2,2}$</td>
<td>$b$</td>
</tr>
<tr>
<td>$p_{2,3}$</td>
<td>$d$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Navigation Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
</tr>
<tr>
<td>$a$</td>
</tr>
<tr>
<td>$b$</td>
</tr>
<tr>
<td>$c$</td>
</tr>
<tr>
<td>$d$</td>
</tr>
</tbody>
</table>
Test Case Generation

I - Model Inference

Page Abstraction → Enrichment & Clustering → Refinement

II - Behavioral Patterns Extraction

Propagation Chains → Navigation Patterns → Singleton

III – Test Cases Generation

Test Cases

IV – Test Cases Execution

Execution
Attack Pattern-based Test Case Generation
Attack Pattern-based Test Case Generation
Test Case Execution and Oracle

I - Model Inference

II - Behavioral Patterns

III – Test Cases Generation

IV – Test Cases Execution
LTL Security Property:

$$\text{ord}_{\text{placed}} \land \text{onStore}(S) \implies 0(\text{paid}(U, I) \land \text{toStore}(S) \land 0(\text{ack}(U, I) \land \text{onStore}(S)))$$
Oracle

HTTP conversation → Events extraction → App. logic checker → Verdict

store/login

```html
<HTML>
<a href="/view?id=23"/>
</HTML>
```

store/view?id=23

```html
<HTML>
<a href="/add?id=23"/>
</HTML>
```

store.com/add?id=23

```html
<HTML>
<a href="/checkout"/>
</HTML>
```

Login

View 23

Add 23

\[ \Pi \vdash \phi \]

Verdict: yes

Verdict: no
Case Study: Shopping Cart Web Applications

Customers

Buy

Shopping Cart Web Apps

Pay

Cashier-as-a-Service
(former Payment Gateways)
## Excerpt of Results

<table>
<thead>
<tr>
<th>Applications</th>
<th>CaaS</th>
<th># Test Cases</th>
<th># TC Exec.</th>
<th>Violations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td># Bugs</td>
</tr>
<tr>
<td>AbanteCart</td>
<td>Std</td>
<td>233</td>
<td>74</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Exp</td>
<td>343</td>
<td>240</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>386</td>
<td>210</td>
<td>126</td>
</tr>
<tr>
<td>Magento</td>
<td>Exp</td>
<td>173</td>
<td>140</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>135</td>
<td>71</td>
<td>30</td>
</tr>
<tr>
<td>OpenCart</td>
<td>Exp</td>
<td>165</td>
<td>117</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>225</td>
<td>128</td>
<td>34</td>
</tr>
<tr>
<td>osCommerce</td>
<td>Exp</td>
<td>137</td>
<td>85</td>
<td>-</td>
</tr>
<tr>
<td>PrestaShop</td>
<td>Exp</td>
<td>302</td>
<td>238</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>224</td>
<td>115</td>
<td>24</td>
</tr>
<tr>
<td>TomatoCart</td>
<td>Exp</td>
<td>600</td>
<td>347</td>
<td>313</td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>222</td>
<td>127</td>
<td>108</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3145</strong></td>
<td><strong>1892</strong></td>
<td><strong>849</strong></td>
<td><strong>60</strong></td>
</tr>
</tbody>
</table>
## Vulnerabilities

<table>
<thead>
<tr>
<th>Application</th>
<th>Popularity</th>
<th>Shop for free</th>
<th>Pay less</th>
<th>No agreement on price</th>
<th>Session Fixation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbanteCart</td>
<td>21,200</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Magento</td>
<td>3,130,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenCart</td>
<td>9,710,000</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>osCommerce</td>
<td>80,500</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>PrestaShop</td>
<td>650,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TomatoCart</td>
<td>119,000</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CS-Cart</td>
<td>260,000</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
osCommerce and AbanteCart: Shopping for Free

<table>
<thead>
<tr>
<th>U₁</th>
<th>Store A</th>
<th>Store B</th>
<th>PayPal</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>login</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>add item I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>checkout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>accₐ, amtₐ, invₐ, URLₐ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>login</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>add item I’</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>checkout</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>accₖ, amtₖ, invₖ, URLₖ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>accₐ, amtₐ, invₐ, URLₐ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>authorize payment to accₐ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>return to URLₖ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>URLₖ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>order placed in Store B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- ack(Uᵢ, I), onStore("Store A")
- ack(Uᵢ, I), onStore("Store B")
- paid(Uᵢ, I’), toStore("Store A")
- ord_placed, onStore("Store B")
Security Testing → Model Checking

- Passive model inference
- Extraction of workflow and data flow patterns
- Attack pattern-based test case generation
- Assessment of real-world web applications
- Detection of 10 critical vulnerabilities that affect millions of websites
Missing Pieces

- Replace attack patterns with Model Checking
  - Our models are still too complex
  - We need a good attacker model for the Web

- The oracle needs to be provided by the analyst
  - Explore inference techniques to generate logic invariants

- Extend the technique to other:
  - classes of vulnerabilities (e.g., improper authentication)
  - classes of applications (e.g., online banking, travel booking, conference systems, ...)