Logic Vulnerabilities in eCommerce Web Applications

Giancarlo Pellegrino, Ph.D. student
giancarlo.pellegrino@{eurecom.fr, sap.com}

OWASP EU Tour 2013
Eurecom, Sophia-Antipolis France
About me

- PhD student at Eurécom, at Software and System Security group

- Working as research associate at SAP AG
  - Contributing to the EU funded project SPaCioS “Secure Provision and Consumption in the Internet of Services”

- Interests in web security, security testing, browser-based security protocols, and formal analysis

- More info:
  - http://trouge.net/gp
What will we talk about?

- Logic Vulnerabilities
- Detection techniques
- Workflow and data flow manipulations in eCommerce Web Applications
What are logic vulnerabilities?
Logic Vulnerability

• …or design flaws/errors, business/application logic errors/flaws

• Logic Vulnerability lacks of formal definition
  – CWE-ID 840: Business Logic Error are “Weaknesses […] that commonly allow attackers to manipulate the business logic of an application.”

• Mainly caused by insufficient validation of the business process workflow and data flow

• Logic Vulnerabilities can exhibit patterns, e.g.
  – Information Exposure
  – Improper Authorization
Information Exposure: MacWorld Expo 2007

- Annually trade-show on Mac platforms
  - e.g., in 2007 Jobs unveiled the first iPhone

- Cost of “Platinum Pass” 1,695$
  - seat to see Jobs up close included

- Discount available through submission of discount codes

- Web page contained MD5 hashes and JavaScript code for client-side validation

- Codes were uppercase, 5-char long strings

Improper Authentication: Business Wire 2005

• Business Wire® a press release company

• Workflow:
  1. upload news, e.g.:
     o http://website/press_release/24/06/2013/1.html
     o http://website/press_release/24/06/2013/2.html
  2. link the news to the user page
  3. access to the news granted only to authenticated users

• Release temporarily interrupted to avoid stock market influences
  – implemented by delaying step 2

• Lohmus Haavel & Viisemann, an Estonian company:
  – registered as user
  – wrote a spider
  – made ~$8 mil by trading the unpublished news

Further examples...

... can be found here:

“Get Rich or Die Trying”

by Jeremiah Grossman
BlackHat 2009
How to detect them?
Black-box: Web Scanners

- Crawl the application, create and execute attacks, analyze the responses

- Mainly focus on injections vulnerabilities, e.g., XSS, SQLi, …

- Logic vulnerabilities out of the scope:
  - No notion of internal state
White-box: Source Code Analysis

- Requires source code for building the model
- Depends on the programming language
- Academic publications:
  - “Toward Automated Detection of Logic Vulnerabilities in Web Applications”, V. Felmetsger, L. Cavedon, C. Kruegel, G. Vigna
Manual testing

- (so far) logic vulnerabilities discovered manually

- OWASP Testing Guide 3.0 suggests:
  1. Understand the web application
     - Intended workflow and data flow
  2. Design tests violating the intended workflow and data flow
     - E.g., reorder steps, replay tokens, …
  3. Run tests and observe the result

- Little support for identifying “interesting” data values to tamper with:
  - “How to Shop for Free Online: Security Analysis of Cashier-as-a-Service Based Web Stores”, R. Wang, S. Chen, X. Wang, S. Qadeer
Workflow and Data Flow manipulations
eCommerce Web Application

- Customers
  - Buy
  - Redirections
  - Pay

- eCommerce Web Application
- Payment Gateway

Payment Gateways:
- PayPal
- Google Checkout
- Sagepay
- Amazon Payments
Example 1: PayPal Payments Standard & Instant Payment Notification

- Add item I
- Ok
- Checkout
- Payee=Seller, amount=$value(I)$, return URL=URLSeller
- Payee=Seller, amount=$value(I)$, return URL=URLSeller
- Payee=Seller, Paid=$value(I)$
- Return to URLSeller
- Return to Seller
- Order placed

Payee=Seller, amount=$value(I)$, return URL=URLSeller

Payee=Seller, Paid=$value(I)$
Example 1: Data flow
Example 1: Design (Can I pay myself?)

Add item I
Ok
Checkout
Payee=Seller, amount=value(I), return URL=URLSeller
Payee=Buyer, amount=value(I), return URL=URLSeller
Payee=Buyer, Paid=value(I)
Return to URLSeller
Return to Seller
Order placed
Example 1: Execution and Assessment

1. **Add item I**
2. **Ok**
3. **Checkout**
4. Payee=Seller, amount=value(I), return URL=URLSeller
5. Payee=Buyer, amount=value(I), return URL=URLSeller
6. **Order placed**
7. **Return to URLSeller**
8. **Status order: payment pending**
9. **Return to Seller**
10. **Payee=Buyer, Paid=value(I)**
11. **Status order: paid**
12. **Order placed**
13. **Return to URLSeller**
14. **Shop for free**
Example 2: PayPal Express Checkout

Buyer

Add item I

Ok

Checkout

Token

PayPal

Start checkout session

Token

Shipping data, payment data

Token, PayerID

Get shipping details

Details

Complete payment

Done

Order placed

Token, PayerID
Example 2: Workflow and Data flow

**Workflow**

- Item selection
- Cart confirmation
- Payment
- Payment confirmation
- Purchase confirmation

**Data flow**

- Add item
- Ok
- Checkout
- Start checkout session
- Token
- Shipping data, payment data
- Get shipping details
- Details
- Complete payment
- Done
- Order placed
- Token
- PayerID
- Token
- Token
- Token
- Token
- PayerID
Example 2a: Design (No payment)

Workflow

Data flow

Item selection

Cart confirmation

Payment

Payment confirmation

Purchase confirmation

Add item

Ok

Checkout

Start checkout session

Token

Shipping data, payment data

Get shipping details

Details

Complete payment

Done

Order placed
Example 2a: Design (No payment)

User Session 1

1. Add Candy
2. Ok
3. Checkout
4. Token1
5. Token1

6. Shipping data, payment data
7. Token1, PayerID
8. Token1, PayerID
9. Order placed

User Session 2

1'. Add Ferrari
2'. Ok
3'. Checkout
4'. Token2
8. Token1, PayerID
9'. Order placed
Example 2a: Execution and Assessment

User Session 1
1. Add Candy
2. Ok
3. Checkout
4. Token1
5. Token1
6. Shipping data, payment data
7. Token1, PayerID
8. Token1, PayerID
9. Order placed

User Session 2
1'. Add Ferrari
2'. Ok
3'. Checkout
4'. Token2
8. Token1, PayerID
9'. Order placed

Status order Candy: payment pending
Status order Candy: paid
Status order Ferrari: payment pending
Status order Ferrari: paid

Pay less
Example 2b: Design (No payment and no confirmation)

User Session 1
1. Add Candy
2. Ok
3. Checkout
4. Token1

User Session 2
1'. Add Ferrari
2'. Ok
3'. Checkout
4'. Token2
5. Token1
6. Shipping data, payment data
7. Token1, PayerID
8. Token1, PayerID
9'. Order placed
Example 2b: Execution and Assessment

User Session 1

1. Add Candy
2. Ok
3. Checkout
4. Token1
5. Token1
6. Shipping data, payment data
7. Token1, PayerID

Status order Candy: payment pending

User Session 2

1'. Add Ferrari
2'. Ok
3'. Checkout
4'. Token2
8. Token1, PayerID
9'. Order placed

Status order Ferrari: payment pending

Status order Ferrari: paid

Shop for free
Summing up

- Logic vulnerabilities
  - Mainly caused by insufficient validation of workflow and data flow

- Hard to discover

- Detection done manually

- Simple manipulation of the workflow and data flow have a great impact
Questions?

Contact information:

Giancarlo Pellegrino
giancarlo.pellegrino@{eurecom.fr, sap.com}
References

- “Get Rich or Die Trying”, by Jeremiah Grossman, presentation at BlackHat 2009
- OWASP Testing Guide v.3.0
- “How to Shop for Free Online: Security Analysis of Cashier-as-a-Service Based Web Stores”, R. Wang, S. Chen, X. Wang, S. Qadeer
- “Toward Automated Detection of Logic Vulnerabilities in Web Applications”, V. Felmetsger, L. Cavedon, C. Kruegegl, G. Vigna