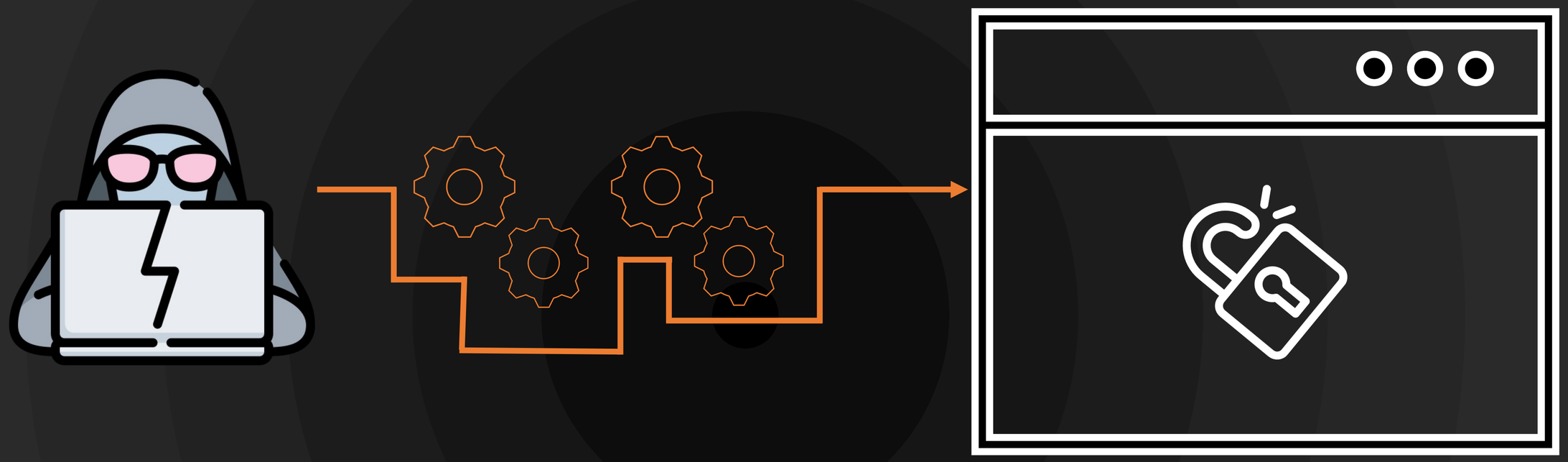


Business Logic Vulnerabilities

Author: Rana Khalil (@rana__khalil)

Agenda

WHAT ARE BUSINESS LOGIC VULNERABILITIES?

HOW DO YOU FIND AND EXPLOIT THEM?

HOW DO YOU PREVENT THEM?

WHAT ARE BUSINESS LOGIC VULNERABILITIES?

Business Logic Vulnerabilities are flaws in the design and implementation of an application that allows an attacker to elicit unintended behavior.

Example 1 – Change Another User's Password

Functionality

The application has a password change for end users and administrators.

- End users need to fill out the username, existing password, new password and confirm new password fields.
- Administrators only need to fill out the username, new password and confirm new password fields.

Assumption

The client-side interface presented to users and administrators is different but the password change is controlled for both users by the same function.

Example 1 – Change Another User's Password

Code

```
String existingPassword = request.getParameter("existingPassword");
if (null == existingPassword) {
  trace("Old password not supplied, must be an administrator");
  return true;
}
else
{
  trace("Verifying user's old password");
...
```

Attack

A regular user submits a request to change another user's password by simply not supplying the existing password.

Example 2 – Bypass Checkout Functionality

Functionality

The application has a “place an order” functionality that follows the following stages:

- Browse the product catalog and add items to the shopping basket.
- Return to the shopping basket and finalize the order.
- Enter the payment.
- Enter delivery information.

Assumption

The developers assumed that users would always access the stages in the intended sequence.

Example 2 – Bypass Checkout Functionality

Attack

The user proceeds directly from stage 2 to stage 4, finalizing the order for delivery without paying for the order.

- Browse the product catalog and add items to the shopping basket.
- Return to the shopping basket and finalize the order.
- Enter the payment.
- Enter delivery information.

Example 3 – Beating a Business Limit

Functionality

A banking application allows users to transfer funds between bank accounts. As a precaution against fraud, the application prevents users from transferring a value greater than $10,000.

Assumption

The developers put a check in place to ensure that no transaction greater than $10,000 is allowed to go through.

```
bool CAuthCheck::RequiresApproval(int amount) {
     if (amount <= m_apprThreshold)
          return false;
     else return true; }
...
```

Example 3 – Beating a Business Limit

Attack

The developers overlooked the possibility that a user would attempt to process a transfer for a negative amount. Any negative number would clear the approval test because it is less than the threshold value.

Therefore, a user who wants to transfer $20,000 from account A to account B could simply initiate a transfer -$20,000 from account B to account A bypassing the antifraud defense.

Example 4 – Cheating on Bulk Discounts

Functionality

An e-commerce website allows users to order software products and qualify for bulk discounts if a suitable bundle of items was purchased. The following are the steps involved in the bulk discount functionality:

1. User adds items in basket.
2. If one of the items qualifies for a bulk discount, a discount is applied on the entire cart.
3. User purchases order.

Assumption

Users will purchase the chosen bundle after the discount is applied.

Example 4 – Cheating on Bulk Discounts

Attack

User can exploit this logic flaw by performing the following steps:

1. User adds items in basket including item that gives the user a bulk discount.
2. The discount is applied on the entire cart.
3. User goes back to the cart and removes the item that entitled him to a discount.
4. Although the item is removed, the discount is still approved, and the user purchases the order at a discounted price.

Impact of Business Logic Vulnerabilities

- The impact is highly variable and depends on the functionality that contains the business logic flaw.
  - Confidentiality – Access to other users' data.
  - Integrity – Access to update other users' data
  - Availability – Access to delete users and their data.

OWASP Top 10

OWASP
Open Web Application Security Project

OWASP Top 10 - 2013	OWASP Top 10 - 2017	OWASP Top 10 - 2021
A1 – Injection	A1 – Injection	A1 – Broken Access Control
A2 – Broken Authentication and Session Management	A2 – Broken Authentication	A2 – Cryptographic Failures
A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure	A3 - Injection
A4 – Insecure Direct Object References	A4 – XML External Entities (XXE)	A4 – Insecure Design
A5 – Security Misconfiguration	A5 – Broken Access Control	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Security Misconfiguration	A6 – Vulnerable and Outdated Components
A7 – Missing Function Level Access Control	A7 – Cross-Site Scripting (XSS)	A7 – Identification and Authentication Failures
A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization	A8 – Software and Data Integrity Failures
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities	A9 – Security Logging and Monitoring Failures
A10 – Unvalidated Redirects and Forwards	A10 – Insufficient Logging & Monitoring	A10 – Server-Side Request Forgery (SSRF)

HOW TO FIND AND EXPLOIT BUSINESS LOGIC VULNERABILITIES?

How to Find & Exploit Business Logic Vulnerabilities

- Map the application. Make note of each and every component in the application and how it operates.
  - If you have access to the code, review the code responsible for each component.
- For each component determine:
  - The potential business flow.
  - The assumptions that could have been made by the developers / architects during the design phase.
- Test each component for all possible use cases that are outside of the intended business flow.

Exploiting Business Logic Labs

LAB APPRENTICE
Excessive trust in client-side controls »

LAB APPRENTICE
High-level logic vulnerability »

LAB APPRENTICE
Inconsistent security controls »

LAB APPRENTICE
Flawed enforcement of business rules »

LAB PRACTITIONER
Low-level logic flaw »

LAB PRACTITIONER
Inconsistent handling of exceptional input »

LAB PRACTITIONER
Weak isolation on dual-use endpoint »

LAB PRACTITIONER
Insufficient workflow validation »

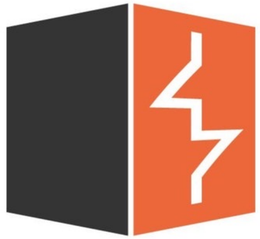
LAB PRACTITIONER
Authentication bypass via flawed state machine »

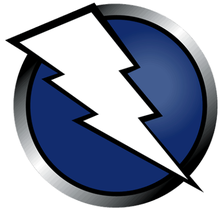
LAB PRACTITIONER
Infinite money logic flaw »

LAB PRACTITIONER
Authentication bypass via encryption oracle »

Automated Exploitation Tools

Web Application Vulnerability Scanners (WAVS)





HOW TO PREVENT BUSINESS LOGIC VULNERABILITIES?

Preventing Business Logic Vulnerabilities

- Ensure that there is proper documentation of the application's design that outlines every assumption that the designer(s) made.
- Mandate that all source code is properly commented and includes the following items:
  - The purpose and intended use of each code component.
  - The assumptions made by each component about anything that is outside of its direct control.
  - References to all client-side code that uses the component.
- Write code as clearly as possible.
- Perform security-focused code reviews of the application's design.

Resources

- Web Security Academy – Business Logic Vulnerabilities
  - *https://portswigger.net/web-security/logic-flaws*
- Web Application Hacker's Handbook
  - *Chapter 11 – Attacking Application Logic*