Exploiting Business Logic Vulnerabilities

Pankaj Verma Follow
Nov 29, 2020 · 3 min read

Business Logic Vulnerabilities in web applications are not new, but these vulnerabilities are extremely varied and too often untested. Testing for business logic flaws in today's multi-functional dynamic web applications requires lateral thinking, systematic probing and unconventional methods.

Hello Fellow Hackers & Security Enthusiasts, I'm back again with a new Bugbounty Writeup. In this article, I'll be sharing some of my recent findings where I've dealt with some Business Logic Flaws in the application. So, let's start with understanding the vulnerability first.

> ***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 behaviour. This potentially enables attackers to manipulate legitimate functionality to achieve a malicious goal.*

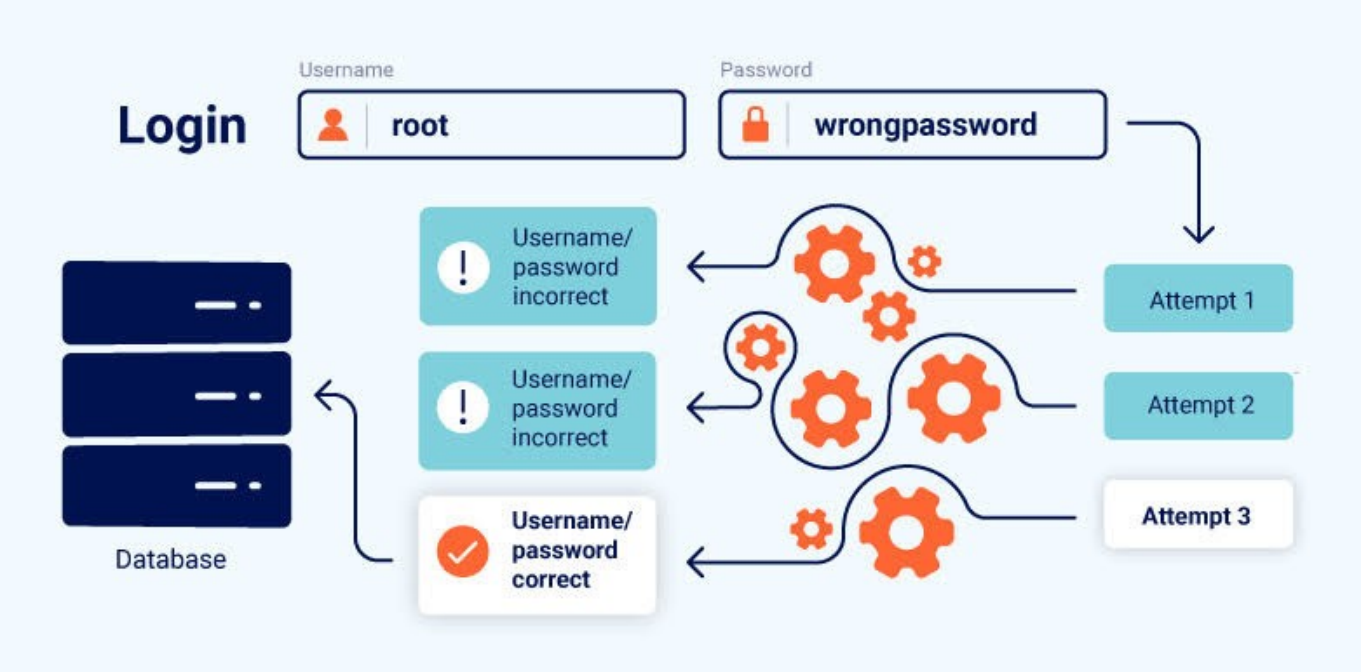

Image Source : PortSwigger

As I always like to hunt for business logic flaws along with technical vulnerabilities because these vulnerabilities carry a high impact on the target application. Recently, I encountered some of the logic-based vulnerabilities while testing a private application.

1. Critical Parameter Manipulation or Logical Data Validation

The application had logical validation at the front end but not on the server-side. The application is verifying data locally that left the application vulnerable to data tampering through Burp proxy or by manipulating client-side code.

The Attack:

The target was an E-Commerce application which was storing the prices of the products within hidden form fields so that one can not edit the price from the front end. But at the time of Checkout, an attacker was able to manipulate the product's price by tampering the request with Burp Suite.
The interesting part of this vulnerability was that the application was accepting the product's price in Negative values also.

```
POST /AJAX HTTP/1.1
Host: target.com
Cookies: session=xxxxxxxx

place_order=true&selected_payment_method=0&last_wallet_money_used=0&l
ast_payable_amount=-999&charged_cod=0&selected_address=xxxxx
```

So, when the attacker submitted a **negative** amount as price, the order gets placed as well as this amount reflects back to the attacker's account wallet as positive credit.

```
HTTP/1.1 200 OK
Array
(
 [wallet_credited_amt] => 999
 [wallet_debited_amt] => 0
 [total_displaying_in_wallet] => 999
)
Order_Number : 185xxxx <br>Successfully Placed and All Orders Details
Added -
```

2. Coupon Code Reuse

The application was using Coupon Code functionality for providing a discount to new users. The intended behaviour was to provide the discount once per user, but the user was able to use the same coupon code for multiple orders which eventually lead to the financial loss of the organisation.

The Attack:

The application was providing Coupon Code to users for availing discounts once per account at the time of checkout. Here, the application was not validating the number of

times a coupon code can be used by the user. So the attacker was able to use the same Coupon Code for as many times as he wants.

```
POST /AJAX HTTP/1.1
Host: target.com
Cookies: session=xxxxxxxx

save_order_summary_for_discount=true&selected_payment_method=0&last_w
allet_money_used=0&last_payable_amount=599&charged_cod=0&discount_cod
e=CC100&selected_address=xxxxx
```

Takeaways :

- Understand how the application works by following its workflow.
- Analyze the application's logic to get all interesting inflows.
- Want to Learn and Practice more about Business Logic Vulnerabilities, Try **The Web Application Hacker's Handbook** & **PortSwigger**.

If you enjoyed reading my article do clap and follow on Medium and Twitter:

Twitter: https://twitter.com/_p4nk4j

LinkedIn: https://www.linkedin.com/in/p4nk4jv/