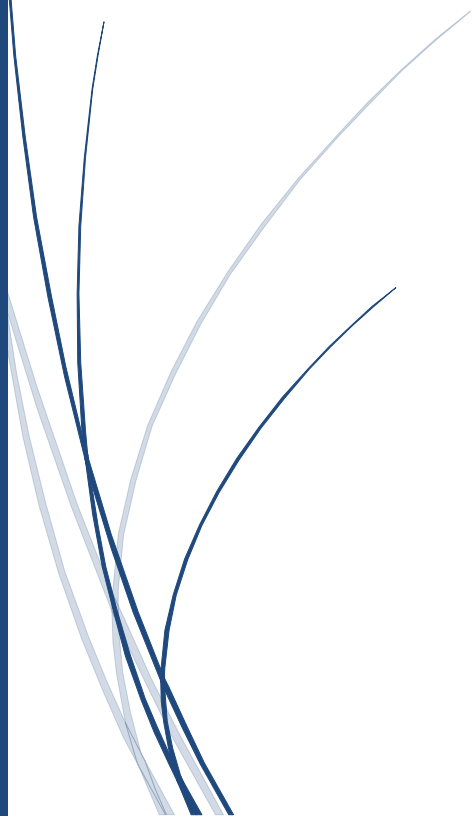
qSEAp
pioneering innovation...

Breaking the Business Logics

- Manas Harsh



Introduction:

This document is intended to provide the idea of business logic vulnerabilities and how to exploit them. There are theoretical scenarios as well where common flaws are discussed.

Description:

Business logic vulnerabilities are flaws in the design and implementation of an application that allows an attacker to bypass the intended behavior. These flaws enable attackers to manipulate the functionalities of a web app to achieve the results they want. With these bugs, an attacker may be able to force the application to work in such a way, which developers didn't intend to.

If we talk about one main purpose of business logics, it would be enforcing the rules of an app which are defined at the time of app development or when the developers create the app functionalities. Business rules are something that defines how an application will work on the given scenarios.

On the other side, flaws in business logics can allow attackers to play with these rules. We will see some methods ahead where business logics could be manipulated and an attacker can get the desired results after manipulating them.

Why there are flaws in business logics:

Business logic vulnerabilities arise because of the developers as they already make some assumptions of how a user will use the application. They don't figure out the second side where user input could be anything and hence it leads to limited validation of user input. Let's assume a scenario where developers assume that users will pass their inputs via web browser and in this case, the application will rely on only client-side controls which are weak and could be bypassed easily by an attacker using an intercepting proxy tool. In simple words, when an attacker performs anything other than intended user behavior, the application fails to prevent it which leads to logic flaws.

Why business logic flaws are critical:

Since business logics have a broad category so the impact differs. However, any unintended behavior can potentially lead to high-severity attacks if an attacker is able to manipulate the application in the right way. The impact of any logical flaw depends on how that particular functionality works. For example, if the flaw is in the authentication mechanism, this could lead to a serious problem with overall application security. An attacker can exploit this for attacks like privilege escalations or they can bypass the authentication as well, which could lead to sensitive data access. Also, it increases the attack surface for an attacker. Moreover, even if business logics are not affecting the app directly, it could still do some damage to business in some other ways.

Where to check them:

Here, we will see the business logics as a pentester. So, if we talk about an application that could have business logic flaws in common, they would be most probably e-commerce websites since they own a lot of functionalities to play with.

However, there could be any website with some functionalities which leads to business logic flaws. Let's assume there is an application that allows us to purchase the home appliance. Here, we can check for the common business logics like if we can reduce the price of an item from our cart. Also, can we apply a coupon code more than once? These are some of the scenarios. Also, if the website has 2FA(2-Factor-authentication) functionality, we can check them as well. Moving forward, we will see some vulnerabilities related to business logics.

Some common vulnerabilities to check:

Price manipulation:

If an application has a functionality to add items to the cart, we can add multiple items there and once we checkout, we have to pay the total amount. Here, we can check if we can manipulate the price of items when we checkout. There are multiple ways but we will discuss the most common way to do that. We will use burp as our intercepting proxy and in most of the cases, the result will look something like this:

items=5&&price=USD100

Here, we can modify the amount and send the request from burp. It will look something like this:

items=5&&price=USD1

Now, if there is a misconfigured business logic in the application, it will accept our value. Now the value we have to pay is USD1.
Also, I have seen some scenarios where the application accepted the negative value i.e., -100 USD and in this case, we can get the money back in our wallet. Sounds weird but it happens in real-life scenarios even though applications are more secure these days.

Coupon code reuse:

Some applications give discounts to us based on the coupon code. Let's assume there is an application where the new registered user can claim a 50% off on his first five purchases. There is a coupon code called FIRST50 and once we apply this, we get 50% off on items.
Here, what we can do is, instead of using this code once, we can try using it multiple times. For a practical scenario, we will simply send a lot of request to the server with the same coupon code. So, if there is a flaw, we will get 50% off every time on the current price.

Currency manipulation:

This is one more business logic flaw that is uncommon but when it works it really harms the business a lot.
What if we could manipulate the currency and convert a USD into INR?
That is a clear business impact. For example, let's assume the price of an item is 100 USD and if we convert it into INR, it would be something like 1.4 USD.
We can see a huge difference in price here.

A practical scenario will be looking something like this:

item=1&&price=100USD

Now, if we can manipulate the price to a different currency i.e., INR, the request will be sent to the server in this format:

item=1&&price=100INR

With this flaw, we can simply reduce the price of an item a lot.

Authentication bypasses:

This is also a critical flaw in business logics where we can bypass the authentications like 2-Factor-authentication. Once an attacker bypasses the authentication, he can simply own someone else's account and completely take it over. We will see some scenarios here related to authentication bypasses:

- An attacker can bypass the authentication if the application doesn't validate the steps and since the developers assume that a user will simply follow the steps as per application, it is easy to manipulate them. To validate this issue, we need to simply create two accounts in an app and login with the first account. Since the app will ask the 2FA code to login, we provide it and log in to the app. Now, we make a note the of URL and logout from the application. We login with another account and this time when app asks for a 2FA, we simply manipulate the URL path to the previous user which we have copied earlier. It bypasses the authentication in a simple way.

- In some cases, application has some broken 2-factor-authentication and we can brute-force the code to get access to someone else's account. For example, if an application sends a 4-digit code on email for authentication, we can simply capture the request and brute-force it with the help of an intruder which is a Burpsuite functionality. If we try to login with someone's email and if brute-force works than we can easily bypass the 2FA and own the account.

Broken password reset functionality:

This business logic flaw occurs when an application doesn't validate the password reset token and hence, we can change the password for anyone. Even though this flaw is very rare these days, it always makes sense to go ahead and check this. To validate this issue, we can create two accounts in an application and from the first account, we try to reset the password as we go to the "forgot password" page. Here, we get an email for resetting the password. However, some apps don't validate the reset token which is sent with the email. So, we use burp as our intercepting tool and capture the request after entering the username and password. Here, we simply remove the parameters such as "temp-reset-token" and check what's the response. If it doesn't validate the token, we can simply change the username and try to send a request. If there is a flaw in business logic, it will change the password for someone else.

Prevention:

There are a few main points that could be implemented for a safe business logic implementation:

- Developers who make the application and functionalities should understand the application properly and domain as well.
- Developers should avoid making assumptions about what a user can do with the functionalities and what not.
- Testers should check all the functionalities if they are working properly in different scenarios.

Along with these points, the application should be checked properly multiple times. Codes should be written clearly. These shouldn't include any complex codes because if a business logic flaw arises, it will be difficult to monitor them if the code is complex.

References:

https://portswigger.net/web-security/logic-flaws
https://hdivsecurity.com/docs/risks/business-logic-flaws/
https://blog.rapid7.com/2015/06/04/top-10-business-logic-attack-vectors/