SECURITYBOAT
Frontline Of Your Business

Insecure Direct Object Reference

HANDBOOK

www.securityboat.net

SECURITYBOAT
Frontline Of Your Business

Table of contents

SECURITYBOAT
Frontline Of Your Business

Let us deep dive into some basics to understand IDOR:

Authentication: Authentication determines or proves the person is "who they say they are" by providing credentials (i.e., username & password), biometric verification or by any other authentications method.

Authorization: Authorization determines what type of access you have for an application or in short it describes your role in a particular application.

Why IDOR Vulnerabilities Arises:

IDOR vulnerability is a type of access control related vulnerabilities.

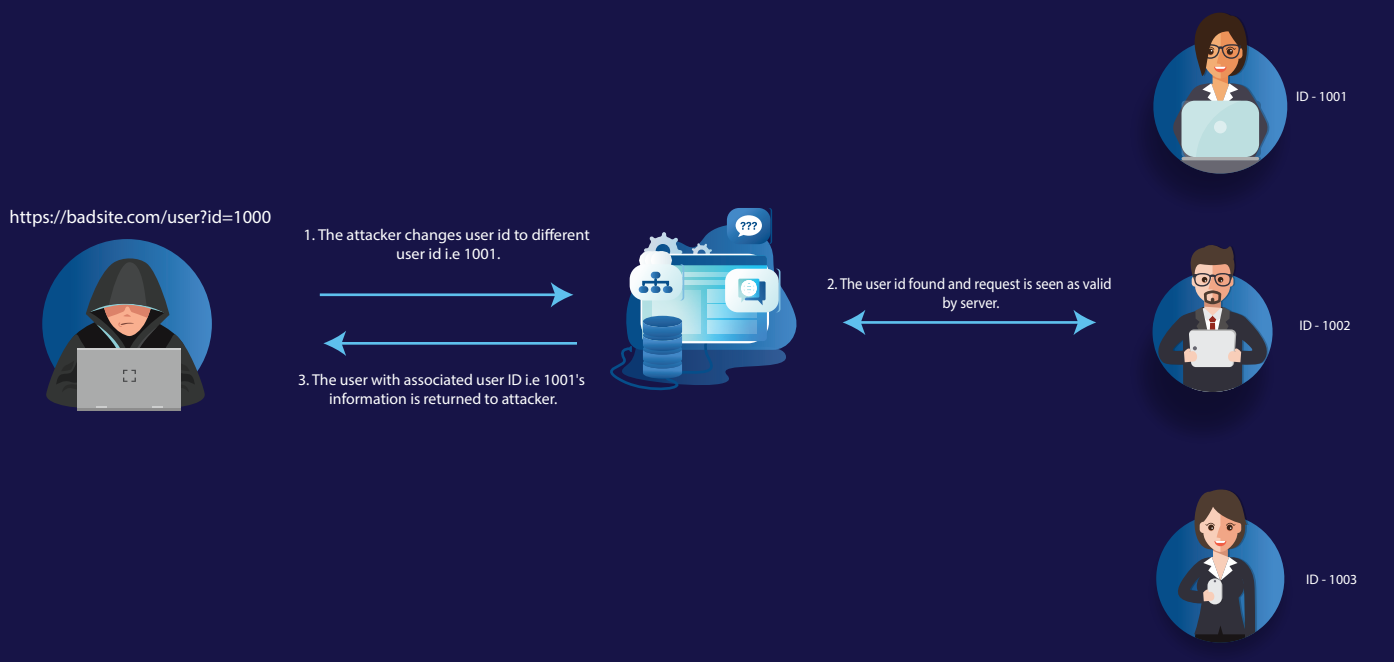
It occurs when applications are not properly sanitizing or validating user input, which can allow attackers to manipulate inputs to gain un-authorized access.

In short, we can say if an application leaks the user account details or can perform actions on behalf of users without properly authorizing the user then there may be chances to exploit IDOR vulnerability there.

Definition of IDOR

IDOR is an abbreviation for Insecure Direct Object Reference, a vulnerability that occurs when an application exposes sensitive information or functionality by referencing an internal object, such as a database record, file, or user account details, without properly validating the user's authorization to access that object.

How IDOR Works:



1. A attacker is requesting for his/her account details that are associated with id=1000

2. Attacker intercepts the request via Burp suite/OWASP ZAP and modifies user id 1000 to user id 1001.

3. The server checks that is there any user available with associated id I.e., 1001.

4. Server checks the id & as it seems valid to server then it returns the details of requested user id to an attacker.

5. As there were no proper authorization checks due to which server was showing the information of another user to an attacker which is causing IDOR.

Let us understand it more by an example:

Let us say there is an application that allows user to view and edit their personal information but what if there is no proper input validation on IDs (unique identifier) which is passing in url & through which you can use sequential IDs to access other user's account.

```
# Code to retrieve user data by ID from a database
def get_user_data(user_id):
    query = "SELECT * FROM users WHERE id = %s"
    cursor.execute(query, (user_id,))
    return cursor.fetchone()

# Code to display user profile page
def display_profile (user_id):
    user_data = get_user_data(user_id)
```

Likewise, in the above snapshot you can see using user_id, application is getting the user data from database as a response. So just by changing the user_id parameter to a different value, the attacker could potentially view the profile data of another user without proper authorization.

```
https://badsite.com/profile?user_id=1234
```

How to checks for IDOR:

Here are some steps to look for while checking IDOR:

Examine the application's source code where objects are being referenced directly without the necessary user authorization checks being made.

Capture the request into proxy tool such as Burp Suite or OWASP ZAP. You can try to access or manipulate sensitive things that you are not authorized to access by fiddling with the requests.

Use Brute-force methods to access the items that are not supposed to be accessible. Let us say if a user can access their own account information by navigating to https://example.com/user/1234, try changing the number at the end to see if you can access other users' accounts.

Examine the application's features and look for places where user input can be utilized to directly reference objects to perform manual testing. To see if you can access confidential information or tamper with the system, try accessing those objects.

Types of IDOR:

Object-level IDOR: When an attacker can directly access and manipulate a specific object, such as a user's account, by changing the object's ID/sequential ID's or other parameters in the application's request or response, this is referred to as object-level (Sequential) IDOR.

JSON-based IDOR: This kind of vulnerability occurs when an application uses JSON data to represent objects. An attacker could utilize JSON data manipulation to gain access to or change sensitive objects if the application does not properly check user input when parsing JSON data.

Suppose a web application uses JSON to transmit data between the client and server, and there is an endpoint that allows users to retrieve information about their own orders. The endpoint might look like this:

https://example.com/orders/1234

In the JSON response, the order information might look like this:

```
{
    "order_id": 1234,
    "customer_id": 5678,
    "order_total": 100.00,
    "items": [

        {

            "item_id": 1,
            "item_name": "Product A",
            "item_price": 50.60
},
{
            15
            "item id": 2,
            "item_name": "Product B",
            "item_price": 56.00
        }
```

An attacker could modify the customer_id value in the JSON response to access information about another user's orders. For example, by changing the customer_id value to 5679, the attacker could access the order information for that user:

Function-level IDOR: This happens when an attacker bypasses authentication procedures by directly accessing an API endpoint or function only meant for authorized users.

Consider the possibility that a web application has an endpoint for account deletion. An attacker can change the request to remove other users' accounts if the website does not adequately authenticate the user's authority to carry out this operation.

Indirect-IDOR: This happens when an attacker can change or indirectly access an object by taking advantage of a flaw in the access controls of the application, such as through guessing or brute-forcing IDs or other parameters.

For example, if a user can view their own profile information at https://example.com/user/johndoe, an attacker could try accessing other users' profiles by guessing their usernames or email addresses via Brute force, then it will be Indirect IDOR (Insecure Direct Object Reference) by guessing the credentials.

Impact of IDOR vulnerabilities/ What we can be achieved through IDOR :

Failures of Access Control leads to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits. Let us see below in more detailed manner:

Data leakage: When an attacker gain access to the victim's account using IDOR then there will be chances of leaking the data.

Authentication Bypass (Broken Access Control): Attacker can simply bypass the authentication mechanism by manipulating/tampering the Id's.

Data Alteration: An attacker may have access to and alter your data. As a result, an attacker may have access to your data and manipulate records. In some cases, the attacker will delete/impersonate the victim's account.

Account Takeover: While an attacker may have multiple access points to user accounts simply by changing the "UID" values, this will result in an account takeover vulnerability. When one vulnerability causes another (as in this case),

Privilege Escalation: When a hacker exploits an access flaw and gains access to a privileged account (normal user to other privilege account like admin or super admin) and its resources, then it is classified as Vertical Privilege escalation.

If a user gain access to the same level of account, then it is classified as Horizontal Privilege Escalation.

Reputational Damage: As we saw in data leakage, if an attacker leaks the data, then it will be reputational damage for the organization. The organization will lose the trust of their customers.

SECURITYBOAT
Frontline Of Your Business

Bypasses of IDOR:

BYPASSES

1. Wrap the ID with an array.

```
POST /api/get_profile HTTP/1.1
Host: example.com
...
{"user_id":111}
```

Try this to bypass

```
POST /api/get_profile HTTP/1.1
Host: example.com
...
{"id":[111]}.
```

2. Wrap the ID with a JSON object

```
POST /api/get_profile HTTP/1.1
Host: example.com
...

{"user_id":111}
```

Try this to bypass

```
POST /api/get_profile HTTP/1.1
Host: example.com
...

{"user_id":{"user_id":111}}
```

3. Add .json to the endpoint

```
GET /v2/GetData/1234 HTTP/1.1
Host: example.com
...
```

Try this to bypass

```
GET /v2/GetData/1234.json HTTP/1.1
Host: example.com
...
```

4. Send wildcard instead of ID

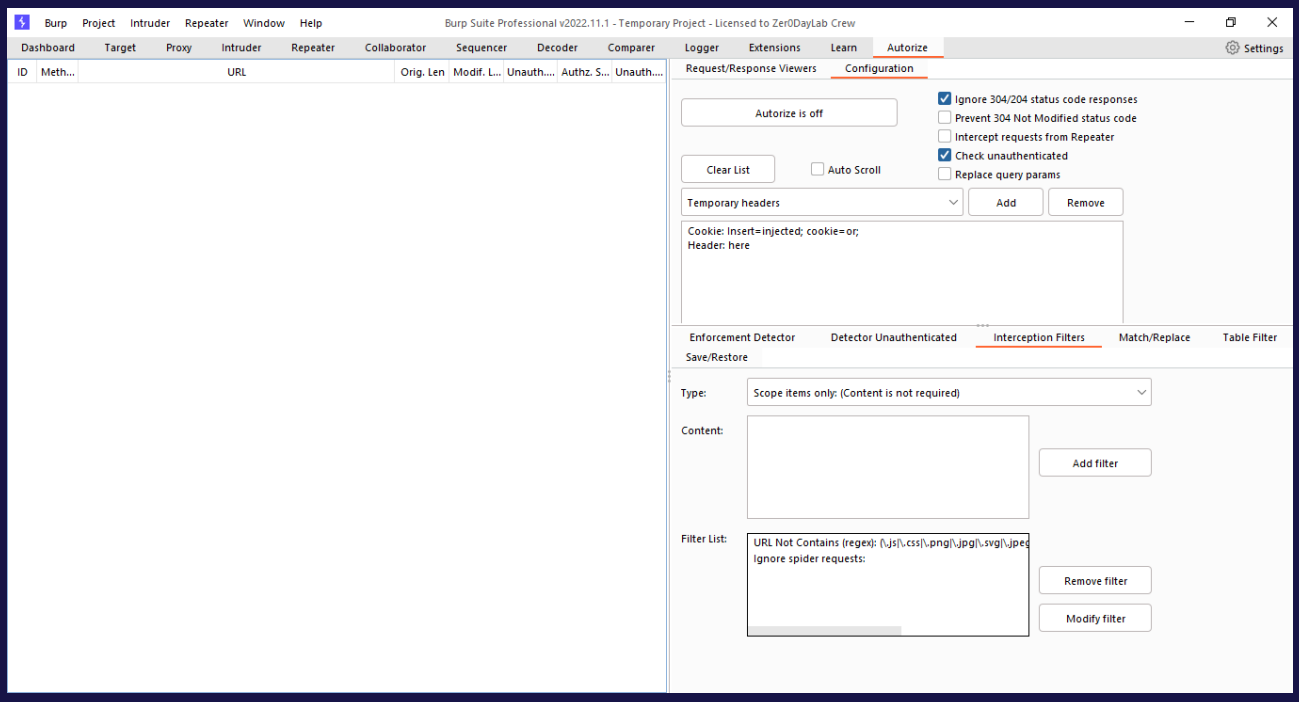
```
GET /api/users/111 HTTP/1.1
Host: example.com
```

Try this to bypass

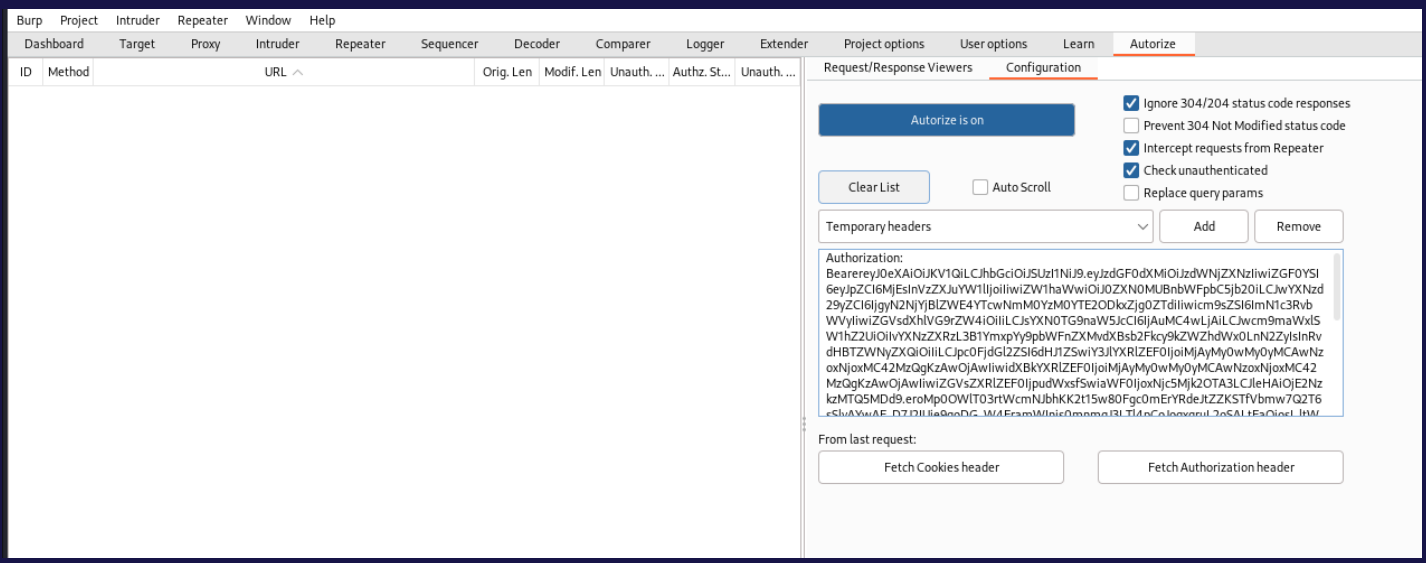
```
GET /api/users/* HTTP/1.1
Host: example.com
```

Automation Tool Autorize

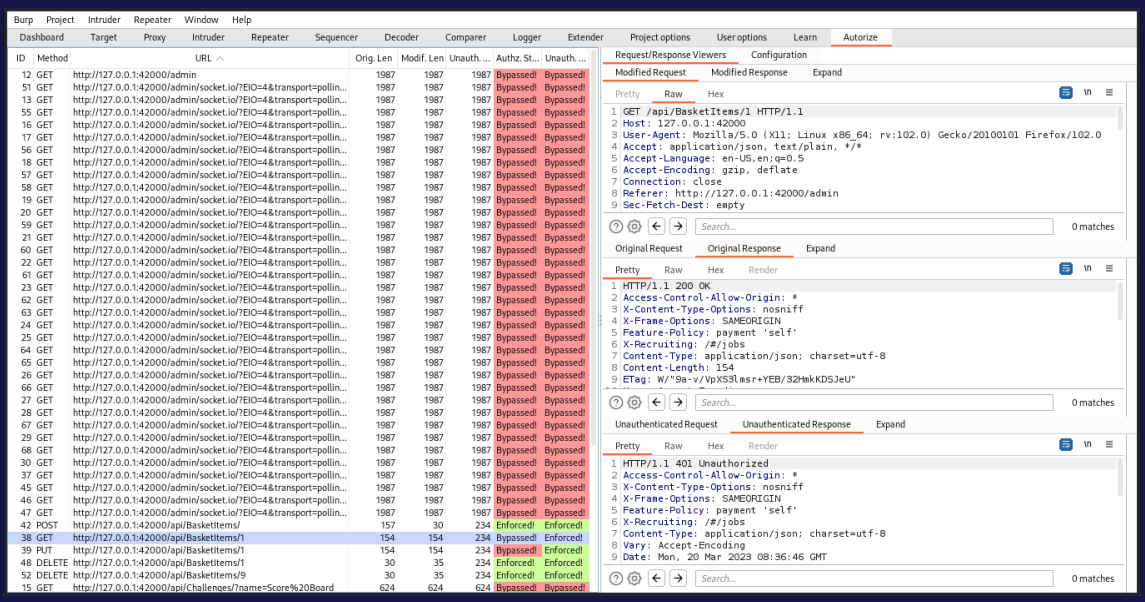
Autorize is a burp suite extension which can be added easily from the BApp Store. This extension automates your penetrating testing for IDOR to much extent. All you need to do is take some headers such as cookies and authorization token of the low privileged user and configure them in the extension. Then go through the modules of other user or high privileged user and the extension will list down if the modules contain potential IDOR in it or not, which you can further check by analyzing the response.



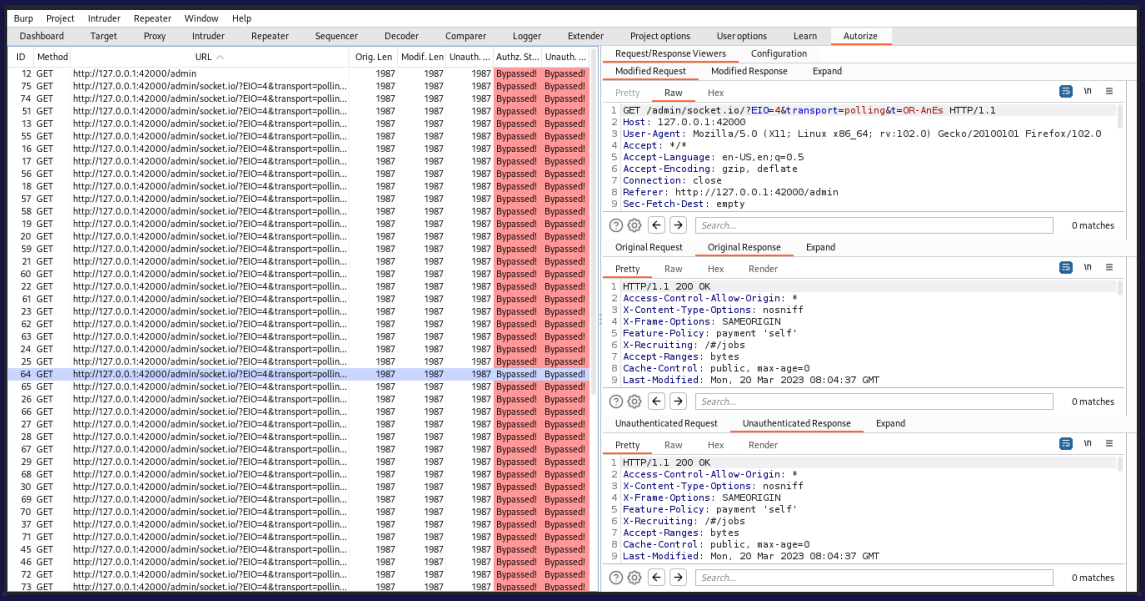
Let us consider an example of vulnerable application juice-shop. First, we login as an administrator and a normal user. We take the authorization and cookies header of the low privileged user and place it in authorize tool as shown below.



After we visit the functionalities of the admin we get two types of responses. The enforced and bypassed. Enforced which is in green color means that the original response and modified response are different and the endpoint is protected from IDOR as shown below.



Bypassed which is in red color means that the original response and modified response are same and there is an IDOR vulnerability as shown below.



There is also a third response called “is enforced” which is in orange color and means that the endpoint seems to be protected and we should take a closer look into it.

IDOR Prevention

Role Based Access Control

IDOR can be prevented by implementing strict Access Control methods to ensure that the user is authorized to access the particular functionality. Permissions such as read, write, and delete should be assigned to users according to their role.

Use indirect references.

A unique token should be used to reference the index instead of directly referencing the object that is indirect references can be used to access objects instead of direct references.

Monitor Continuously

The logs should be continuously monitored to check for potential IDOR attacks.

Perform Security Testing

Regular audits should be conducted to find IDORs before they are exploited by an attacker. Security Testing such as vulnerability scanning, penetrating testing should be performed, and vulnerabilities should be patched on time.

Use Parameterized Queries

By using parameterized query, it is ensured by the developer that the input is sanitized and validated which prevents vulnerabilities such as IDOR.

Summary

IDOR (Insecure Direct Object Reference) is a serious security vulnerability that can have profound consequences for an organization if not handled properly. It enables an attacker to access sensitive data or perform actions on behalf of another user without proper authorization. This can result in data breaches, financial loss, and reputational harm. Organizations must take proactive measures to prevent IDOR vulnerabilities in their applications. This can be accomplished by implementing appropriate access controls, input validation, and session management. Additionally, regular security audits and testing are required to identify and correct any potential vulnerabilities.

To summarize, preventing IDOR vulnerabilities necessitates a multi-dimensional approach that includes both technological and educational solutions. Organizations can reduce the risk of IDOR vulnerabilities and ensure the security and privacy of their users' data by implementing best practices and staying vigilant.

References:

https://book.hacktricks.xyz/pentesting-web/idor

https://github.com/daffainfo/AllAboutBugBounty/blob/mater/Insecure%20Direct%20Object%20References.md

https://medium.com/cyberverse/automating-burp-to-find-idors-2b3dbe9fa0b8

SECURITYBOAT
Frontline Of Your Business

Insecure Direct Object Reference

HANDBOOK

Scan QR Code to Download Handbook