Intro to IDOR

`Insecure Direct Object References (IDOR)` vulnerabilities are among the most common web vulnerabilities and can significantly impact the vulnerable web application. IDOR vulnerabilities occur when a web application exposes a direct reference to an object, like a file or a database resource, which the end-user can directly control to obtain access to other similar objects. If any user can access any resource due to the lack of a solid access control system, the system is considered to be vulnerable.

Building a solid access control system is very challenging, which is why IDOR vulnerabilities are pervasive. In addition, automating the process of identifying weaknesses in access control systems is also quite difficult, which may lead to these vulnerabilities going unidentified until they reach production.

For example, if users request access to a file they recently uploaded, they may get a link to it such as (`download.php?file_id=123`). So, as the link directly references the file with (`file_id=123`), what would happen if we tried to access another file (which may not belong to us) with (`download.php?file_id=124`)? If the web application does not have a proper access control system on the back-end, we may be able to access any file by sending a request with its `file_id`. In many cases, we may find that the `id` is easily guessable, making it possible to retrieve many files or resources that we should not have access to based on our permissions.

What Makes an IDOR Vulnerability

Just exposing a direct reference to an internal object or resource is not a vulnerability in itself. However, this may make it possible to exploit another vulnerability: a `weak access control system`. Many web applications restrict users from accessing resources by restricting them from accessing the pages, functions, and APIs that can retrieve these resources. However, what would happen if a user somehow got access to these pages (e.g., through a shared/guessed link)? Would they still be able to access the same resources by simply having the link to access them? If the web application did not have an access control system on the back-end that compares the user's authentication to the resource's access list, they might be able to.

There are many ways of implementing a solid access control system for web applications, like having a Role-Based Access Control (RBAC) system. The main takeaway is that `an IDOR vulnerability mainly exists due to the lack of an access control on the back-end`. If a user had direct references to objects in a web application that lacks access control, it would be possible for attackers to view or modify other users' data.

Many developers ignore building an access control system; hence, most web applications and mobile applications are left unprotected on the back-end. In such applications, all users may have arbitrary access to all other user's data on the back-end. The only thing stopping users from accessing other user's data would be the front-end implementation of the application, which is designed to only show the user's data. In such cases, manually manipulating HTTP requests may reveal that all users have full access to all data, leading to a successful attack.

All of this makes IDOR vulnerabilities among the most critical vulnerabilities for any web or mobile application, not only due to exposing direct object references but mainly due to a lack of a solid access control system. Even a basic access control system can be challenging to develop. A comprehensive access control system covering the entire web application without interfering with its functions might be an even more difficult task. This is why IDOR/Access Control vulnerabilities are found even in very large web applications, like Facebook, Instagram, and Twitter.

Impact of IDOR Vulnerabilities

As mentioned earlier, IDOR vulnerabilities can have a significant impact on web applications. The most basic example of an IDOR vulnerability is accessing private files and resources of other users that should not be accessible to us, like personal files or credit card data, which is known as `IDOR Information Disclosure Vulnerabilities`. Depending on the nature of the exposed direct reference, the vulnerability may even allow the modification or deletion of other users' data, which may lead to a complete account takeover.

Once an attacker identifies the direct references, which may be database IDs or URL parameters, they can start testing specific patterns to see whether they can gain access to any data and may eventually understand how to extract or modify data for any arbitrary user.

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

IDOR vulnerabilities may also lead to the elevation of user privileges from a standard user to an administrator user, with `IDOR Insecure Function Calls`. For example, many web applications expose URL parameters or APIs for admin-only functions in the front-end code of the web application and disable these functions for non-admin users. However, if we had access to such parameters or APIs, we may call them with our standard user privileges. Suppose the back-end did not explicitly deny non-admin users from calling these functions. In that case, we may be able to perform unauthorized administrative operations, like changing users' passwords or granting users certain roles, which may eventually lead to a total takeover of the entire web application.

Previous Next Mark Complete & Next