A learning path for Broken Access Control Vulnerabilities

Broken access control was ranked as 1st in the 2021 OWASP top 10 list.
Over 500,000 applications were tested for some form of broken access control with the average incidence rate of 3.81% with a maximum of 55.97% and have the most occurrences in the contributed dataset with over 318k. [1]

Beginner Level

Access control policies grant or restrict users' rights on the application ensuring that users cannot act outside their intended permissions. Therefore, these techniques manage resource permissions of the application with the purpose of avoiding users' actions to cause potential security problems.
In the context of web applications, access control relies on authentication and session management. Authentication is used to identify the user and to ensure that he is who he claims he is. Session management binds HTTP requests to the user and access control determines whether the user is allowed to perform the actions he is attempting to perform. [2]

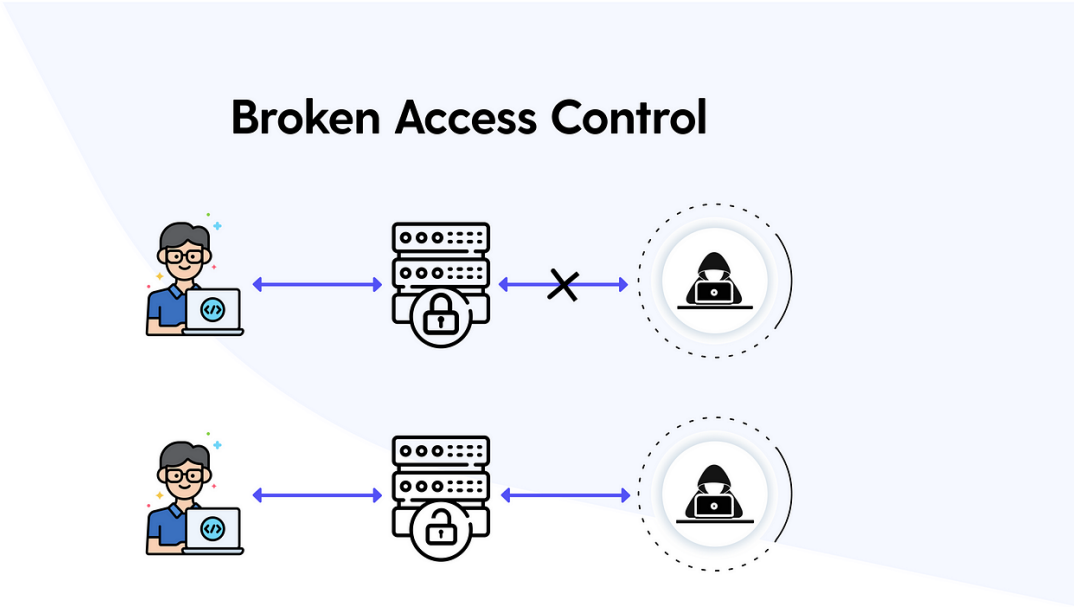

Fig. 2.1 Simple Broken Access Control Schema [3]

If the access control is implemented in the right way, only authorized users can perform some specific actions, while malicious users cannot. On the other hand, if there are some access control issues, an attacker could manage to bypass control policies and perform actions without any permission.

Depending on which resources the attackers get the permissions to access, the impacts of a successful attack could be detrimental. Failures in access control usually lead to modification or destruction of sensitive data, unauthorized information disclosure or performing a business function outside the user's limit. [1]

One of the CWEs included in the broken access control category is CWE-352: Cross-Site Request Forgery (CSRF).

A Cross Site Request Forgery is an attack that tricks the victim into submitting a malicious HTTP request to a target destination without their knowledge or intent, in order to perform an action disguised as the victim.

“When a web server is designed to receive a request from a client without any mechanism for verifying that it was intentionally sent, then it might be possible for an attacker to trick a client into making an unintentional request to the web server which will be treated as an authentic request. This can be done via a URL, image load, XMLHttpRequest, etc. and can result in exposure of data or unintended code execution.” [4]

Therefore, this attack can be performed by exploiting the trust that a website has for a user; in fact, it can work on sites in which the user is currently authenticated.

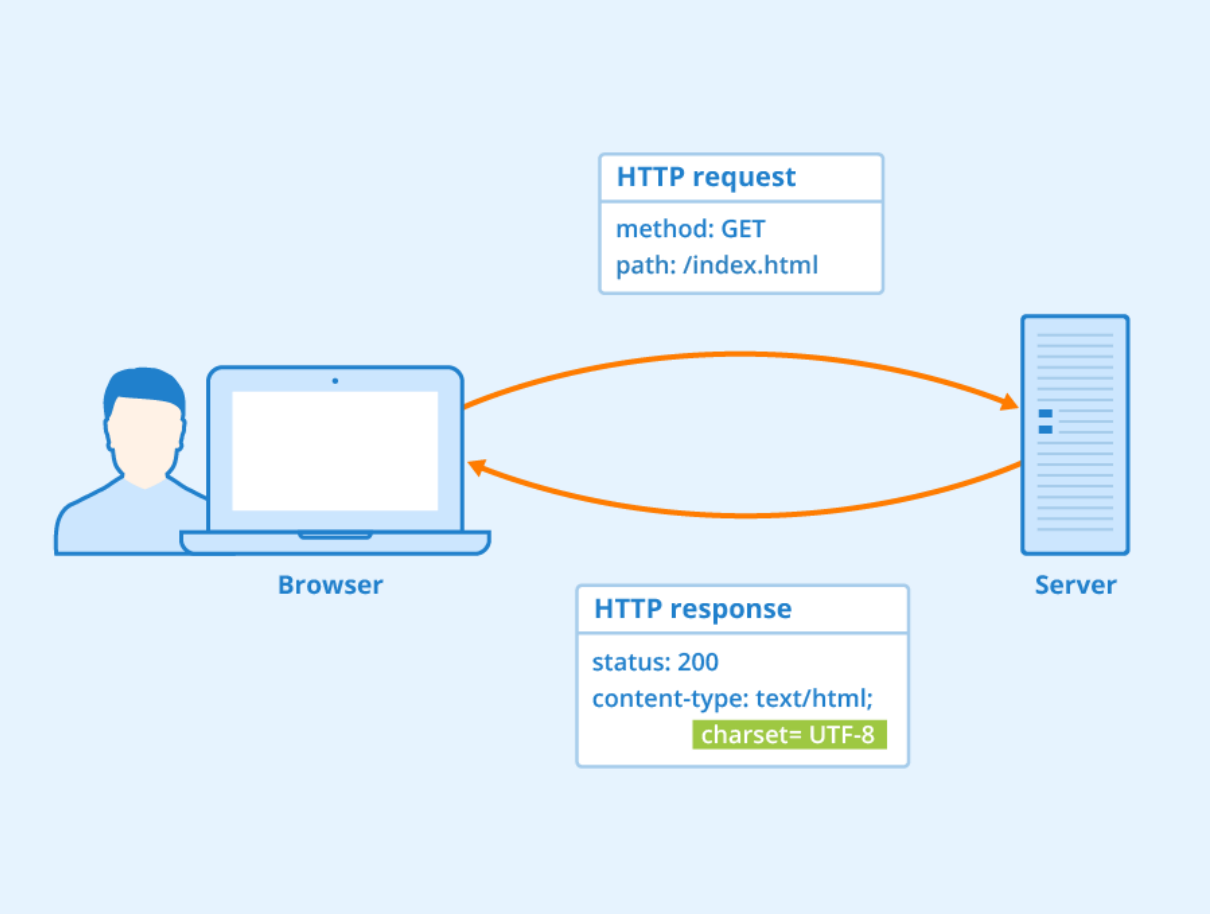

Fig. 2.2 Example of HTTP GET request [5]

In order to have a better understanding of this attack, it is necessary to understand how a request can be generated.

The request (in the *Fig. 1.8* case a GET request) could be sent by the user in several different ways: by using the web application, by typing the URL directly in the browser or by following an external link that points to the URL.
All these kinds of invocations are indistinguishable by the server and this could be very dangerous. Particularly, the third one is the most dangerous.
In fact, there are various different ways to disguise the real properties of a link. The link can be embedded in an email message, appear in a malicious website to which the user is attracted to or appear in content hosted by third-parties and point to a resource of the application. If the user clicks on the link, since he has already been authenticated by the web application, the browser will issue a GET request accompanied by user's authentication information (for example a session ID cookie). This results in a valid operation being performed on the web application that the user does not expect. For example, it might be changing the email associated with the account, changing the password or a fund transfer on a web banking application.
Therefore, the attacker could manage to gain complete access over the account. Depending on user's privileges in the compromised application, the attacker could also be able to take control over the entire application data and functionality.

A CSRF attack always involves three actors: a trusted site, a victim user of the trusted site and a malicious site. The victim user simultaneously visits the malicious site while holding an active session with the trusted site.

Bibliography

[1] «A01:2021 – Broken Access Control,» [Online]. Available: https://owasp.org/Top10/A01_2021-Broken_Access_Control. [Consulted 2023].

[2] «Access control vulerabilities and privilege escalation,» [Online]. Available: https://portswigger.net/web-security/access-control. [Consulted 2023].

[3] [Online]. Available: https://miro.medium.com/v2/1*3z3zxJj_GzArgT2SIy529w.png.

[4] «CWE-352: Cross-Site Request Forgery (CSRF),» [Online]. Available: https://cwe.mitre.org/data/definitions/352.html. [Consulted 2023].