Broken Access Control

White Hats: Team Members
Banks Cargill - cargillb@oregonstate.edu
Frederick Eley - eleyf@oregonstate.edu
Timothy Glew - glewt@oregonstate.edu

Description:

Another web application security risk listed by the OWASP Foundation is broken access control. Access control is also commonly referred to as authorization. Once a user is authenticated and logged into a web application, restrictions on what data and resources that user may access should be implemented. Failure to secure access can lead to the exposure of sensitive data which may be viewed, modified, or deleted. An attacker may even be able to take control of an entire system if they gain administrative rights.

Access controls are divided into three different categories; vertical, horizontal, and context-dependent, each having its own vulnerabilities.

Vertical access controls allow different types of users to access different data and functions in an application. This is often seen in the difference between administrators and regular users. An administrator typically has the ability to modify or delete accounts. If a regular user were to gain unauthorized access to an admin page they would not only be able to view and/or download all users' data, they would also be able to modify or delete the accounts making them inaccessible to the original users. Vertical privilege escalation often occurs when administrative data and functions are at a specific URL that contain no authorization checks. For example, an administrative panel might be at the URL http://foo-bar.com/admin. A developer may only have a link to this page visible when an administrator logs in and presume that it is secure because only authenticated admins will see this link. The page is actually very insecure as anyone with knowledge of this URL can now access the administrative panel. The URL is visible on-screen and may be seen by shoulder surfers. The URL may be stored in the browser history or proxy logs. Administrators may even write down, bookmark, or email this URL. Finally, the location of an administrative URL may be discovered by reading a web application's robots.txt file or by brute forcing a URL with a wordlist.

Horizontal access controls allow users to access data and resources of the same type. For example, an online store will allow a user to see the items in their shopping cart but not those of another user. Horizontal privilege escalation occurs when a user is able to gain access to data and resources belonging to another user. Horizontal privilege escalation often occurs by exploiting URL parameters as well. For example a user may log in to an account page with the URL http://foo-bar.com.com/account?id=123. In an application with no access controls, simply changing the id parameter could allow access to another user's account. Specific accounts may be targeted particularly when the URL parameters are not integers but user IDs such as an account for Jane Doe at http://foo-bar.com.com/account?id=doej. The problem here lies in that user IDs are accessible through means outside the web application such as email messages.

Context-dependent access controls restrict access to functionality and resources according to the state of the application or the user's interaction. This prevents a user from performing actions out of order. For example, an online store prevents users from modifying the contents of their shopping cart after they have made payment. Again context-dependent privilege escalation can occur by URL exploitation but in a slightly different way. An application may store access information in a hidden field, cookie, or preset query string parameter. For example, at checkout a shipping invoice may be filled when a user reaches the URL http://foo-bar.com/shipping.jsp?paid=true, indicating that the payment process has been cleared. Without access controls, a user with knowledge of this URL could bypass the payment page and go directly to the shipping phase.

The best ways to prevent access control vulnerabilities are:

Assume that URL locations will be found either manually or by automation and that access should be granted only through proper authorization.

Unless a resource is intended to be publicly accessible, deny access by default.

Log access control failures and alert admins when on repeated failures.

Limit failed access attempts.

How to Attack our Site:

[Detailed explanation (with screenshots or code snippets) of how to carry out the attack on our site]

How to Defend our Site:

[Detailed explanation (with screenshots or code snippets) of how we have patched the vulnerability]

References