Broken Access Control

Description

Access control, sometimes called authorization, is how a web application grants access to content and functions to some users and not others. These checks are performed after authentication, and govern what ‘authorized’ users are allowed to do. Access control sounds like a simple problem but is insidiously difficult to implement correctly. A web application’s access control model is closely tied to the content and functions that the site provides. In addition, the users may fall into a number of groups or roles with different abilities or privileges.

Developers frequently underestimate the difficulty of implementing a reliable access control mechanism. Many of these schemes were not deliberately designed, but have simply evolved along with the web site. In these cases, access control rules are inserted in various locations all over the code. As the site nears deployment, the ad-hoc collection of rules becomes so unwieldy that it is almost impossible to understand.

Many of these flawed access control schemes are not difficult to discover and exploit. Frequently, all that is required is to craft a request for functions or content that should not be granted. Once a flaw is discovered, the consequences of a flawed access control scheme can be devastating. In addition to

The OWASP® Foundation works to improve the security of software through its community-led open source software projects, hundreds of chapters worldwide, tens of thousands of members, and by hosting local and global conferences.

Upcoming OWASP Global Events

OWASP Global AppSec Singapore 2023

- October 4-5, 2023

OWASP Global AppSec Washington DC 2023

- October 30 - November 3, 2023

OWASP Global AppSec San Francisco 2024

- September 23-27, 2024

viewing unauthorized content, an attacker might be able to change or delete content, perform unauthorized functions, or even take over site administration.

One specific type of access control problem is administrative interfaces that allow site administrators to manage a site over the Internet. Such features are frequently used to allow site administrators to efficiently manage users, data, and content on their site. In many instances, sites support a variety of administrative roles to allow finer granularity of site administration. Due to their power, these interfaces are frequently prime targets for attack by both outsiders and insiders.

Environments Affected

All known web servers, application servers, and web application environments are susceptible to at least some of these issues. Even if a site is completely static, if it is not configured properly, hackers could gain access to sensitive files and deface the site, or perform other mischief.

Examples and References

- OWASP Proactive Controls: Enforce Access Controls
- OWASP Application Security Verification Standard: V4 Access Control
- OWASP Testing Guide: Authorization Testing
- OWASP Cheat Sheet: Authorization Cheat Sheet

How to Determine If You Are Vulnerable

Virtually all sites have some access control requirements. Therefore, an access control policy should be clearly documented. Also, the design documentation should capture an approach for enforcing this policy. If this documentation does not exist, then a site is likely to be vulnerable.

The code that implements the access control policy should be checked. Such code should be well structured, modular, and most likely centralized. A detailed code review should be performed to validate the correctness of the access control implementation. In addition, penetration testing can be quite useful in determining if there are problems in the access control scheme.

Find out how your website is administered. You want to discover how changes are made to webpages, where they are tested, and how they are transported to the production server. If administrators can make changes remotely, you want to know how those communications channels are protected. Carefully review each interface to make sure that only authorized administrators are allowed access. Also, if there are different types or groupings of data that can be accessed through the interface, make sure that only authorized data can be accessed as well. If such interfaces employ external commands, review the use of such commands to make sure they are not subject to any of the command injection flaws described in this paper.

How to Protect Yourself

The most important step is to think through an application’s access control requirements and capture it in a web application security policy. We strongly recommend the use of an access control matrix to define the access control rules. Without documenting the security policy, there is no definition of what it means to be secure for that site. The policy should document what types of users can access the system, and what functions and content each of these types of users should be allowed to access. The access control mechanism should be extensively tested to be sure that there is no way to bypass it. This testing requires a variety of accounts and extensive attempts to access unauthorized content or functions.

Some specific access control issues include:

- **Insecure Id's** – Most web sites use some form of id, key, or index as a way to reference users, roles, content, objects, or functions. If an attacker can guess these id's, and the supplied values are not validated to ensure the are authorized for the current user, the attacker can exercise the access control scheme freely to see what they can access. Web applications should not rely on the secrecy of any id's for protection.
- **Forced Browsing Past Access Control Checks** – many sites require users to pass certain checks before being granted access to certain URLs that are typically 'deeper' down in the site. These checks must not be bypassable by a user that simply skips over the page with the security check.
- **Path Traversal** – This attack involves providing relative path information (e.g., "../../target_dir/target_file") as part of a request for information. Such attacks try to access files that are normally not directly accessible by anyone, or would otherwise be denied if requested directly. Such attacks can be submitted in URLs as well as any other input that ultimately accesses a file (i.e., system calls and shell commands).

- **File Permissions** – Many web and application servers rely on access control lists provided by the file system of the underlying platform. Even if almost all data is stored on backend servers, there are always files stored locally on the web and application server that should not be publicly accessible, particularly configuration files, default files, and scripts that are installed on most web and application servers. Only files that are specifically intended to be presented to web users should be marked as readable using the OS’s permissions mechanism, most directories should not be readable, and very few files, if any, should be marked executable.
- **Client Side Caching** – Many users access web applications from shared computers located in libraries, schools, airports, and other public access points. Browsers frequently cache web pages that can be accessed by attackers to gain access to otherwise inaccessible parts of sites. Developers should use multiple mechanisms, including HTTP headers and meta tags, to be sure that pages containing sensitive information are not cached by user’s browsers.

There are some application layer security components that can assist in the proper enforcement of some aspects of your access control scheme. Again, as for parameter validation, to be effective, the component must be configured with a strict definition of what access requests are valid for your site. When using such a component, you must be careful to understand exactly what access control assistance the component can provide for you given your site’s security policy, and what part of your access control policy that the component cannot deal with, and therefore must be properly dealt with in your own custom code.

For administrative functions, the primary recommendation is to never allow administrator access through the front door of your site if at all possible. Given the power of these interfaces, most organizations should not accept the risk of making these interfaces available to outside attack. If remote administrator access is absolutely required, this can be accomplished without opening the front door of the site. The use of VPN technology could be used to provide an outside administrator access to the internal company (or site) network from which an administrator can then access the site through a protected backend connection.

Edit on GitHub

Spotlight: GitLab

Corporate Supporters

Become a corporate supporter

HOME PROJECTS CHAPTERS EVENTS ABOUT

PRIVACY SITEMAP CONTACT