CRASHTEST **SECURITY**

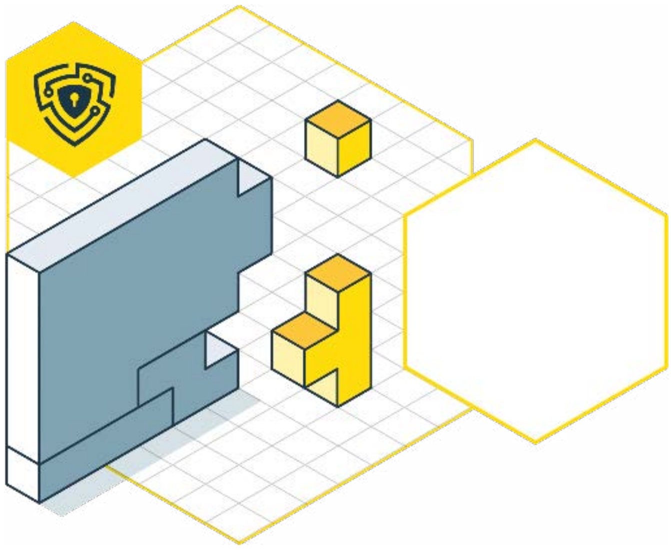
GUIDE FOR

PREVENTING BROKEN ACCESS CONTROL

WHAT ARE THE STEPS TO KEEP YOUR WEB APP OR API SAFE FROM SUCH VULNERABILITY

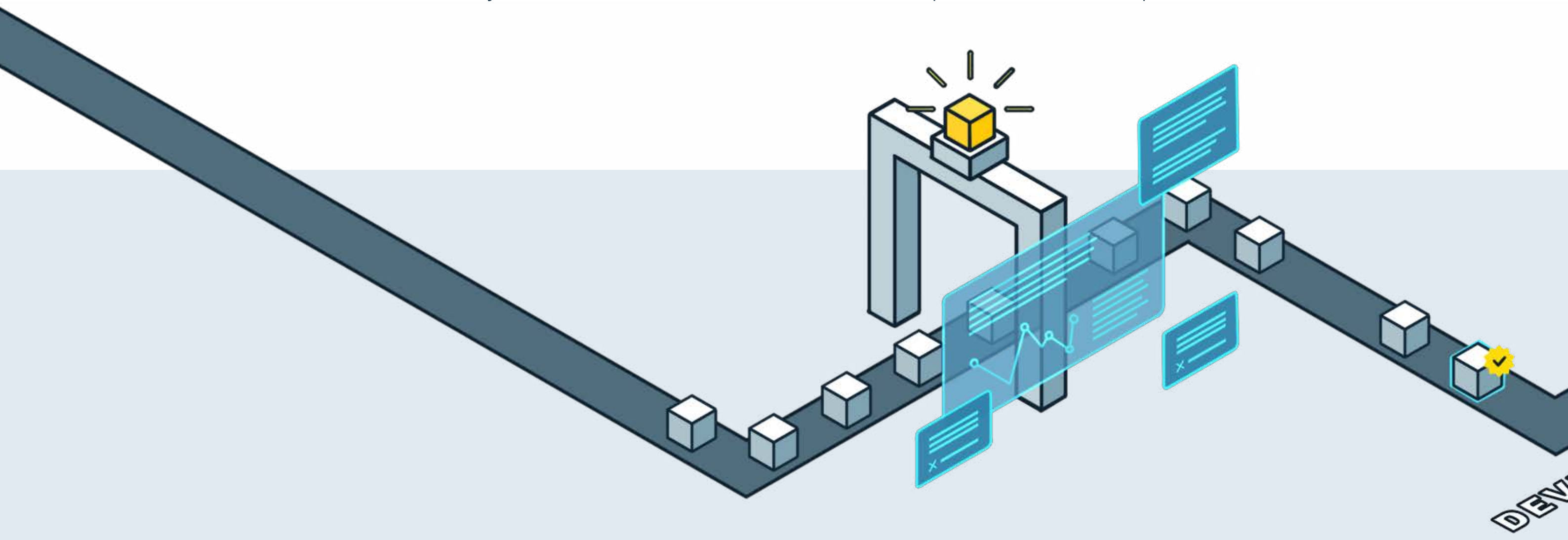
GUIDE FOR BROKEN ACCESS CONTROL PREVENTION

Table of Contents



INTRODUCTION TO THIS GUIDE

Access control is crucial for modern web development as it enables the management of how subjects (users, processes, and devices) should be granted permissions to application functions and resources. Access control mechanisms also determine the level of access permitted and manifest activities carried out by specific entities. Broken access control vulnerabilities arise when a malicious user abuses the constraints on the actions they are allowed to perform or the objects they can access. Attackers typically leverage access control failures to gain unauthorized access to resources within the application, run malicious commands, or gain a privileged user's permission. This guide discusses broken access control vulnerabilities, the severity of associated attacks, and common prevention techniques.



WHAT IS BROKEN ACCESS CONTROL?

Access control issues enable unauthorized users to access, modify and delete resources or perform actions beyond the permissions. Broken access control encompasses various security vulnerabilities typically exploited to elevate privilege levels. Developing secure and effective access control schemes is often a complex undertaking that spans multiple application functions that were not designed deliberately but have evolved with the application. Software developers often overlook how entities access resources when implementing these schemes, resulting in hidden authorization flaws. Such control flaws are typically easy to discover and exploit, making them a popular target for common attacks.

TYPES OF BROKEN ACCESS CONTROL

Broken access control vulnerabilities mostly lead to Privilege Escalation attacks and are characterized by how the malicious user exploits and modifies access rights. The primary forms of access control vulnerabilities include:

HORIZONTAL PRIVILEGE ESCALATION

Horizontal privilege escalation vulnerabilities occur when a user can obtain access to the accounts of other regular users with the same level of permissions. An attacker can leverage these vulnerabilities to get the legitimate user's data and use it for a wide range of malicious acts such as ransomware attacks, financial fraud/unauthorized money transfer, exposure of sensitive files, and data deletion. A horizontal privilege escalation attack usually does not require sophisticated attack tooling and can be orchestrated with a few simple steps, such as:

- By modifying the URL's request ID parameter with legitimate user details obtained through some form of social engineering
- By reviewing the application code to identify authentication vulnerabilities at the source code level
- Using third-party code review tools combined with security testing tools
- Enumerating user accounts on Linux machines

VERTICAL PRIVILEGE ESCALATION

Vertical privilege escalation, also known as privilege elevation, allows an unauthorized user to gain higher privilege levels, typically admin privileges. Privilege elevation usually follows an initial attack, as the malicious user intends to obtain permissions beyond what the compromised subject already has. When compared to horizontal escalation, vertical privilege escalation attacks are more sophisticated since the hacker is required to perform root or kernel-level modifications to obtain administrative access.

Once the attackers gain access rights of admin users, they can inject malicious payloads at the code level, disrupt a sensitive business function or impact the availability of the application‘s critical resources. Some common techniques hackers use to abuse vertical access controls include:

- Using the Windows sysinternals suite to create backdoor administrative users
- Using process injection to mimic administrative functions
- Leveraging directory listing vulnerabilities to disclose information about the access control policy
- Using social engineering for direct access to admin accounts

CONTEXT-BASED PRIVILEGE ESCALATION

A hybrid attack in which the malicious user first obtains access to regular user accounts and then uses broken vertical access controls to gain administrative rights. Context-based privilege escalation attacks also involve business logic exploitation that allows users to perform usually impossible actions within their security context. Examples of context-based privilege escalation include:

- Leveraging Insecure Direct Object Reference vulnerabilities to access critical resources via user-supplied input
- Using corrupt HTTP referer headers for access to functionality and sensitive files beyond their permitted context
- Location-based attacks

BROKEN ACCESS CONTROL - SEVERITY LEVEL

Broken access control is ranked number one on the 2021 OWASP Top 10 security vulnerabilities for web application environments. These vulnerabilities enable attackers to masquerade as different types of users and take control of legitimate user accounts. Depending on the actual vulnerability exploited, the consequences of a privilege escalation attack can be severe.

Specific attack scenarios include:

- Use of insecure IDs - Attackers randomly guess the references for users, roles, objects, contents, and functions. Hackers can easily manipulate access control rules to obtain elevated privilege levels if the vulnerable application does not sanitize the supplied user input.
- Forced browsing - Most applications use multiple security checks to grant access to critical resources within the website‘s backend. Hackers use brute force techniques to bypass the pages running authentication checks, obtaining direct access to web resources.
- Path transversal attacks - The hacker includes a relative path within a URL request, which may grant them direct access to sensitive files.

- Cache attacks - Web browsers store frequently accessed web pages locally within the cache memory. Attackers can obtain cache data and exploit them to replicate administrative functions and orchestrate deeper attacks.
- File permissions - The file permission vulnerability affects files stored on a web server that should not be publicly available. If the server's OS mechanism allows these directories to be readable, attackers can modify application scripts, configuration files, and other default files to cause operational efficiency.

Prevalence of the CSRF vulnerability is considerably common in modern web applications. Attackers also often leverage the fact that most web applications use predictable parameters for actions. However, using appropriate code analysis and penetration testing techniques, the detectability of a CSRF vulnerability is relatively uncomplicated. Since such an identification technique requires **multi-stage delivery of payloads** and is well documented, a CSRF vulnerability's overall severity is typically regarded as of average exploitability.

Impacts of a privilege escalation attack include:

- Takeover of site administration functions
- Modification or deletion of site content
- User account takeover
- Delivery of malicious payloads
- Distributed denial-of-service
- Unauthorized money transfer

Access control vulnerabilities are commonly exploited, with a maximum incidence rate of 55.97% and an average incidence rate of 6.82%. Broken access control vulnerabilities have an average weighted impact of 5.93 and an average coverage rate of 47.72%.

Vulnerabilities associated with broken access control fall under several common weaknesses and enumerations, including:

- CWE-23: Relative Path Traversal
- CWE-59: Improper Link Resolution Before File Access (‚Link Following')
- CWE-201: Exposure of Sensitive Information Through Sent Data
- CWE-219: Storage of File with Sensitive Data Under Web Root
- CWE-275: Permission Issues
- CWE-284: Improper Access Control
- CWE-352: Cross-Site Request Forgery (CSRF)
- CWE-377: Insecure Temporary File
- CWE-402: Transmission of Private Resources into a New Sphere (‚Resource Leak')
- CWE-425: Direct Request (‚Forced Browsing')
- CWE-441: Unintended Proxy or Intermediary (‚Confused Deputy')
- CWE-497: Exposure of Sensitive System Information to an Unauthorized Control Sphere
- CWE-538: Insertion of Sensitive Information into Externally-Accessible File or Directory

IDENTIFYING BROKEN ACCESS CONTROL WITH CRASHTEST SECURITY

Through AI-driven testing and comprehensive vulnerability scanning, Crashtest Security Suite helps generate an in-depth analysis of a tech stack's security and access control. Crashtest Security Suite includes a list of scanners that collectively help analyze broken access control. The list of scanners includes:

- Privilege Escalation Scanner - A vulnerability scanner built to alert admins of any flaws that may lead to abuse of existing access control mechanisms.
- CSRF Scanner - Helps prevent access control attacks using malicious payloads submitted through a trusted normal user.
- URL Fuzzer Scanner - Prevents privilege escalation attacks orchestrated through forced browsing or modifying URL request parameters with a relevant admin URL.
- HTTP Header Scanner - Prevents the use of modified HTTP referer headers to access critical resources beyond the current security context.
- Fingerprinting Scanner - A security scanner used to detect attack surfaces that expose application server implementations, privacy laws, and the web application's access control policy to external domains.

Crashtest Security's automated scanning reduces manual efforts, and lets developers focus quickly on implementing secure design and threat mitigation policies. The platform also offers actionable security reports that can be shared across cross-functional teams, clients and executives, subsequently helping to administer security as a shared responsibility across all verticals of an organization.

BROKEN ACCESS CONTROL PREVENTION TECHNIQUES

Some techniques to prevent access control issues include:

MULTI-FACTOR AUTHENTICATION

Multi-Factor authentication (MFA) is a zero-trust approach to administering security that deploys a series of access control checks that make it difficult for a hacker to perform malicious activities even after acquiring legitimate user credentials. This multi-layered defense strategy combines different authentication mechanisms to validate a user's identity. Mandatory requirement of two or more proofs of identification (such as authentication tokens or biometric IDs) for granting access essentially blocks unauthenticated users from exploiting a normal user account, thereby preventing access control attack attempts.

PROPER AUTHORIZATION SCHEMES

Using CSRF tokens within custom request headers offers a more robust defense mechanism against CSRF attacks as it enforces the same-origin policy restriction. Modern browsers do not support custom headers to be transported with Cross-Domain requests by default. Custom headers can only be added in JavaScript or within the script's origin. This CSRF mitigation technique is stateless and requires no changes to the user experience, making it particularly useful for CSRF mitigation on a REST service.

Apart from considering scalability and agility as the critical features of an application, software developers should build the software with robust access controls and security in mind. Authorization models that enable the creation of effective control units include:

- Discretionary access control - Limiting access based on the subject's identity and/or the groups they belong to. A subject with direct access permission can indirectly assign that permission to other subjects of their choosing.
- Mandatory access control - This mechanism involves securing access to resources based on the sensitivity of the information held by the resource. Administrators label the data sensitivity consisting of a security level and one or more security categories. This allows subjects to only access information held by a resource whose security label applies to them.
- Role-based access control - This access control scheme divides network subjects' access levels into roles. The user is attached to a role, which allows access to the information and functionality needed to perform their duties effectively. Roles are typically built upon job competency, authority, and responsibility.
- Attribute-based access control - Permits or denies information exchange based on properties of the requesting entity, requested action, the context of information exchange, or the resource requested. Some properties used in attribute-based authorization include:
    - Location
    - Threat level evaluation
    - Time of day
    - Security measures implemented on the requested resource

UNIT AND INTEGRATION TESTS

Development teams should implement unit tests at each stage of the software lifecycle to prevent access control flaws at the code level. Unit testing helps evaluate individual modules to ensure appropriate implementation of access control on application code and uncover class-level weaknesses in privilege management. On the other hand, integration tests cover a more extensive scope that includes third-party and open source components used in building the application, thereby helping to evaluate the overall security posture.

SESSION MANAGEMENT

Session management is a critical consideration for building secure software. As such, the appropriate implementation of session IDs, authentication tokens, and cookies collectively prevent session hijacking attacks. Such deployments are provisioned to forcefully destroy session-associated data on an application server after a subject logs out of the application. Implementing session timeouts that require re-authentication and a fresh token when a user connects to the server after logout is also recommended. Designers and developers should also ensure not to expose session IDs in URLs, as attackers could exploit these for session theft techniques.

BEST PRACTICES IN PREVENTING BROKEN ACCESS CONTROL

Some recommended practices to ensure effective access control include:

ENFORCE THE LAW OF LEAST PRIVILEGE

Denying access by default for all public resources is one of the first steps to controlling misuse of access privileges. This implies that each user should be granted permissions required to complete a particular function and no more. The law of least privilege implements a zero-trust approach to information exchange that ensures subjects don't have privileges beyond what is necessary.

WRITING APPLICATION CODE AND BUSINESS LOGIC WITH AUTHORIZATION CONTROLS IN MIND

Software developers and business designers must ensure their program and business logic includes rules that define access to resources and functionalities at the code level. Once the system has authenticated a subject, their privileges to objects should be limited by their roles and identity.

USE CENTRALIZED AUTHORIZATION ROUTINES

Administering access control policies and routines in a centralized location is a recommended approach that helps expedite the application of vulnerability fixes as soon as those are identified. Centralization also eliminates the manual effort required to apply access control policies to every page containing sensitive files and information.

PERFORM SERVER-SIDE CONTROLS

User authentication, input validation, and request processing should be performed at the server-side, as this simplifies the remote management of access control routines. Server-side access control mechanisms also eliminate reliance on traditional keys to enforce privilege decisions, making tracking all access attempts and activities easy.

TEST AND AUDIT ACCESS CONTROLS FREQUENTLY

Apart from manually testing control mechanisms, it is also recommended to adopt automated scanning tools for continuous monitoring of access control flaws that misalign with an organization's security policy. While continuous testing and vulnerability scanning help teams evaluate whether access control mechanisms are working as intended, such tools also help uncover emerging vulnerabilities within access control systems.

DETECTING BROKEN ACCESS CONTROL VULNERABILITIES WITHIN WEB APPS AND APIS WITH CRASHTEST SECURITY

Crashtest Security Suite's vulnerability scanners help establish a continuous, automated security testing process that allows teams to uncover access control flaws with extremely low false positives. The security suite integrates with almost all popular software stacks and security platforms, helping to initiate penetration testing within minutes.

Try Crashtest Security for a free, 14-day demo to understand how the security suite can help eliminate access control blind spots while saving time and budget on blackbox pentesting.

Start 2-Week Trial for Free

CRASHTEST **SECURITY**

WWW.CRASHTEST-SECURITY.COM