[Skip to main content](https://lms.alnafi.com/xblock/block-v1:alnafi+DCCS102+2025_DCCS+type@vertical+block@22f6f549ac8941519f30cb92efddb3d0?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#main)

**Test Defenses Against Application Misuse**

**Summary**

The misuse and invalid use of valid functionality can identify attacks attempting to enumerate the web application, identify weaknesses, and exploit vulnerabilities. Tests should be undertaken to determine whether there are application-layer defensive mechanisms in place to protect the application.

The lack of active defenses allows an attacker to hunt for vulnerabilities without any recourse. The application’s owner will thus not know their application is under attack.

**Example**

An authenticated user undertakes the following (unlikely) sequence of actions:

1. Attempt to access a file ID their roles is not permitted to download
2. Substitutes a single tick ' instead of the file ID number
3. Alters a GET request to a POST
4. Adds an extra parameter
5. Duplicates a parameter name/value pair

The application is monitoring for misuse and responds after the 5th event with extremely high confidence the user is an attacker. For example the application:

* Disables critical functionality
* Enables additional authentication steps to the remaining functionality
* Adds time-delays into every request-response cycle
* Begins to record additional data about the user’s interactions (e.g. sanitized HTTP request headers, bodies, and response bodies)

**How to Test**

This test is unusual in that the result can be drawn from all the other tests performed against the web application. While performing all the other tests, take note of measures that might indicate the application has in-built self-defense:

* Changed responses
* Blocked requests
* Actions that log a user out or lock their account

These may only be localized. Common localized (per function) defenses are:

* Rejecting input containing certain characters
* Locking out an account temporarily after a number of authentication failures

Localized security controls are not sufficient. There are often no defenses against general mis-use such as:

* Forced browsing
* Bypassing presentation layer input validation
* Multiple access control errors
* Additional, duplicated or missing parameter names
* Multiple input validation or business logic verification failures with values that cannot be the result user mistakes or typos
* Structured data (e.g. JSON, XML) of an invalid format is received
* Blatant cross-site scripting or SQL injection payloads are received
* Utilizing the application faster than would be possible without automation tools
* Change in continental geo-location of a user
* Change of user agent
* Accessing a multi-stage business process in the wrong order
* Large number of, or high rate of use of, application-specific functionality (e.g. voucher code submission, failed credit card payments, file uploads, file downloads, log outs, etc).

These defenses work best in authenticated parts of the application, although rate of creation of new accounts or accessing content (e.g. to scrape information) can be of use in public areas.

Not all the above need to be monitored by the application, but there is a problem if none of them are. By testing the web application, and doing the above type of actions, was any response taken against the tester? If not, the tester should report that the application appears to have no application-wide active defenses against misuse. Note it is sometimes possible that all responses to attack detection are silent to the user (e.g. logging changes, increased monitoring, alerts to administrators and request proxying), so confidence in this finding cannot be guaranteed. In practice, very few applications (or related infrastructure such as a web application firewall) are detecting these types of misuse.

**Applications should implement active defenses to fend off attackers and abusers**