[Skip to main content](https://lms.alnafi.com/xblock/block-v1:alnafi+DCCS102+2025_DCCS+type@vertical+block@9affca56f1424149b1bdcf23f08451a6?exam_access=&recheck_access=1&show_bookmark=0&show_title=0&view=student_view#main)

**Summary**

The intrinsic complexity of interconnected and heterogeneous web server infrastructure, which can include hundreds of web applications, makes configuration management and review a fundamental step in testing and deploying every single application. It takes only a single vulnerability to undermine the security of the entire infrastructure, and even small and seemingly unimportant problems may evolve into severe risks for another application on the same server. In order to address these problems, it is of utmost importance to perform an in-depth review of configuration and known security issues, after having mapped the entire architecture.

Proper configuration management of the web server infrastructure is very important in order to preserve the security of the application itself. If elements such as the web server software, the back-end database servers, or the authentication servers are not properly reviewed and secured, they might introduce undesired risks or introduce new vulnerabilities that might compromise the application itself.

For example, a web server vulnerability that would allow a remote attacker to disclose the source code of the application itself (a vulnerability that has arisen a number of times in both web servers or application servers) could compromise the application, as anonymous users could use the information disclosed in the source code to leverage attacks against the application or its users.

The following steps need to be taken to test the configuration management infrastructure:

* The different elements that make up the infrastructure need to be determined in order to understand how they interact with a web application and how they affect its security.
* All the elements of the infrastructure need to be reviewed in order to make sure that they don't contain any known vulnerabilities.
* A review needs to be made of the administrative tools used to maintain all the different elements.
* The authentication systems, need to reviewed in order to assure that they serve the needs of the application and that they cannot be manipulated by external users to leverage access.
* A list of defined ports which are required for the application should be maintained and kept under change control.

After having mapped the different elements that make up the infrastructure (see [Map Network and Application Architecture](https://github.com/OWASP/wstg/blob/master/document/4-Web_Application_Security_Testing/01-Information_Gathering/10-Map_Application_Architecture.md)) it is possible to review the configuration of each element founded and test for any known vulnerabilities.

**Test Objectives**

* Review the applications' configurations set across the network and validate that they are not vulnerable.
* Validate that used frameworks and systems are secure and not susceptible to known vulnerabilities due to unmaintained software or default settings and credentials.

**How to Test**

**Known Server Vulnerabilities**

Vulnerabilities found in the different areas of the application architecture, be it in the web server or in the back end database, can severely compromise the application itself. For example, consider a server vulnerability that allows a remote, unauthenticated user to upload files to the web server or even to replace files. This vulnerability could compromise the application, since a rogue user may be able to replace the application itself or introduce code that would affect the back end servers, as its application code would be run just like any other application.

Reviewing server vulnerabilities can be hard to do if the test needs to be done through a blind penetration test. In these cases, vulnerabilities need to be tested from a remote site, typically using an automated tool. However, testing for some vulnerabilities can have unpredictable results on the web server, and testing for others (like those directly involved in denial of service attacks) might not be possible due to the service downtime involved if the test was successful.

Some automated tools will flag vulnerabilities based on the web server version retrieved. This leads to both false positives and false negatives. On one hand, if the web server version has been removed or obscured by the local site administrator the scan tool will not flag the server as vulnerable even if it is. On the other hand, if the vendor providing the software does not update the web server version when vulnerabilities are fixed, the scan tool will flag vulnerabilities that do not exist. The latter case is actually very common as some operating system vendors back port patches of security vulnerabilities to the software they provide in the operating system, but do not do a full upload to the latest software version. This happens in most GNU/Linux distributions such as Debian, Red Hat or SuSE. In most cases, vulnerability scanning of an application architecture will only find vulnerabilities associated with the "exposed" elements of the architecture (such as the web server) and will usually be unable to find vulnerabilities associated to elements which are not directly exposed, such as the authentication back ends, the back end database, or reverse proxies [1] in use.

Finally, not all software vendors disclose vulnerabilities in a public way, and therefore these weaknesses do not become registered within publicly known vulnerability databases [2]. This information is only disclosed to customers or published through fixes that do not have accompanying advisories. This reduces the usefulness of vulnerability scanning tools. Typically, vulnerability coverage of these tools will be very good for common products (such as the Apache web server, Microsoft's Internet Information Server, or IBM's Lotus Domino) but will be lacking for lesser known products.

This is why reviewing vulnerabilities is best done when the tester is provided with internal information of the software used, including versions and releases used and patches applied to the software. With this information, the tester can retrieve the information from the vendor itself and analyze what vulnerabilities might be present in the architecture and how they can affect the application itself. When possible, these vulnerabilities can be tested to determine their real effects and to detect if there might be any external elements (such as intrusion detection or prevention systems) that might reduce or negate the possibility of successful exploitation. Testers might even determine, through a configuration review, that the vulnerability is not even present, since it affects a software component that is not in use.

It is also worthwhile to note that vendors will sometimes silently fix vulnerabilities and make the fixes available with new software releases. Different vendors will have different release cycles that determine the support they might provide for older releases. A tester with detailed information of the software versions used by the architecture can analyse the risk associated to the use of old software releases that might be unsupported in the short term or are already unsupported. This is critical, since if a vulnerability were to surface in an old software version that is no longer supported, the systems personnel might not be directly aware of it. No patches will be ever made available for it and advisories might not list that version as vulnerable as it is no longer supported. Even in the event that they are aware that the vulnerability is present and the system is vulnerable, they will need to do a full upgrade to a new software release, which might introduce significant downtime in the application architecture or might force the application to be re-coded due to incompatibilities with the latest software version.

**Administrative Tools**

Any web server infrastructure requires the existence of administrative tools to maintain and update the information used by the application. This information includes static content (web pages, graphic files), application source code, user authentication databases, etc. Administrative tools will differ depending on the site, technology, or software used. For example, some web servers will be managed using administrative interfaces which are, themselves, web servers (such as the iPlanet web server) or will be administrated by plain text configuration files (in the Apache case [3]) or use operating-system GUI tools (when using Microsoft's IIS server or ASP.Net).

In most cases the server configuration will be handled using different file maintenance tools used by the web server, which are managed through FTP servers, WebDAV, network file systems (NFS, CIFS) or other mechanisms. Obviously, the operating system of the elements that make up the application architecture will also be managed using other tools. Applications may also have administrative interfaces embedded in them that are used to manage the application data itself (users, content, etc.).

After having mapped the administrative interfaces used to manage the different parts of the architecture it is important to review them since if an attacker gains access to any of them he can then compromise or damage the application architecture. To do this it is important to:

* Determine the mechanisms that control access to these interfaces and their associated susceptibilities. This information may be available online.
* Change the default username and password.

Some companies choose not to manage all aspects of their web server applications, but may have other parties managing the content delivered by the web application. This external company might either provide only parts of the content (news updates or promotions) or might manage the web server completely (including content and code). It is common to find administrative interfaces available from the Internet in these situations, since using the Internet is cheaper than providing a dedicated line that will connect the external company to the application infrastructure through a management-only interface. In this situation, it is very important to test if the administrative interfaces can be vulnerable to attacks.

<https://github.com/OWASP/wstg/blob/master/document/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/01-Test_Network_Infrastructure_Configuration.md>