Vulnerability report

Time of the report : 2023-04-21 18:56:07.365384

We have scanned the entire web application, here is a report summarizing all the vulnerabilities as well as the measures to take to correct these flaws.

List of pages to be tested

['/.', '/instructions.php', '/setup.php', '/vulnerabilities/brute/', '/vulnerabilities/exec/', '/vulnerabilities/csrf/', '/vulnerabilities/fi/.?page=include.php', '/vulnerabilities/upload/', '/vulnerabilities/captcha/', '/vulnerabilities/sqli/', '/vulnerabilities/sqli_blind/', '/vulnerabilities/weak_id/', '/vulnerabilities/xss_d/', '/vulnerabilities/xss_r/', '/vulnerabilities/xss_s/', '/vulnerabilities/csp/', '/vulnerabilities/javascript/', '/security.php', '/phpinfo.php', '/about.php', '/logout.php', '/https://www.virtualbox.org/', '/https://www.vmware.com/', '/https://www.apachefriends.org/en/xampp.html', '/http://www.itsecgames.com/', '/http://sourceforge.net/projects/mutillidae/files/mutillidae-project/', '/http://www.irongeek.com/i.php?page=mutillidae/mutillidae-deliberately-vulnerable-php-owasp-top-10', '/https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project']

Report of detected flaws

Warning ! We discovered a total of 4 flaws !

Vulnerabilities	Numbers
XSS	1
LFI	1
SQL	2

We found one or more XSS vulnerabilities on this link : http://127.0.0.1/https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project, this is the payload used
<script>alert("ok")</script>

We found one or more LFI vulnerabilities on this link : http://127.0.0.1/https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project, this is the payload used
'+UNION+SELECT+@@VERSION,+%27OKAY%27+%23

We found one or more SQL vulnerabilities on this link : http://127.0.0.1/https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project, this is the payload used
../../../../../../../etc/passwd

We found one or more SQL vulnerabilities on this link : http://127.0.0.1/https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project, this is the payload used
../../../../../../../etc/passwd

Recommendation

XSS exploit

XSS injections belong to the category of code injection vulnerabilities in the same way as SQL injections. However, to discover and exploit an XSS vulnerability, an attacker must inject malicious code via the client-side input parameters.

How to defend against it ?

At the PHP code level :
- The XSS attack is due to input coming from the outside (thus untrusted). The solution is therefore to filter the user's input by applying functions such as addslashes() or strip_tags() which remove all tags contained in the input string.
- Only a certain maximum length should be allowed for entries by checking it with the strlen() function or by systematically truncating it with the substr() function.

At the server configuration level :
- Set the magic_quotes_gpc directive in php.ini. This will automatically escape all special characters (including single and double quotes) in strings from outside the system. -> magic_quotes_gpc = On

CSP :
- Apply the Content Security Policy mechanism. CSP (Content Security Policy) script rules must be defined. These allow you to authorise content according to a list that you decide and thus prohibit the loading of specific types of content. You can thus define whether external scripts can be loaded, whether inline scripts can be executed, whether resources can be loaded from the same origin as the main page, etc..

https://capec.mitre.org/data/definitions/63.html
https://www.chiny.me/attaque-xss-cross-site-scripting-14-3.php

LFI exploit

An LFI vulnerability aims to include external files at code execution time. The vulnerability is created because there is not enough checking of the website entries.

How to defend against it ?

At the PHP code level :
- Filtering user input. Similar to the other attacks, the best way in php to prevent this is to filter the inputs. Since the GET parameter passed by the hacker is often too long, you should start by limiting the allowed length.

At the server configuration level :
- By modifying the php engine configuration file called php.ini. This is a text file that contains a set of

directives that allow you to customise certain aspects of the PHP engine's behaviour. magic_quotes_gp can be used to automatically escape special characters from incoming strings (which come from forms, URL bars and cookies). It actually acts like the addslashes function but in an automated way. -> magic_quotes_gpc = On
- There is a directive that allows you to specify whether or not to allow external files to be retrieved for inclusion at runtime. On the php.ini file, you need to disable the following directive -> allow_url_fopen = Off

https://www.invicti.com/web-vulnerability-scanner/vulnerabilities/local-file-inclusion/

SQL exploit

SQL injections belong to the category of code injection vulnerabilities in the same way as XSS injections. It involves adding an unintended query to an SQL query to interact with the database. An SQL flaw allows data to be stolen or modified, or even remote code to be executed.

How to defend against it ?

At the PHP code level :
- Filtering user input. Anything the user may submit to the server is considered foreign and potentially hostile content. Therefore, before allowing input to pass, it must be filtered.One can simply escape special characters from submitted strings using PHP functions like addslashes() or mysql_real_escape_string(). One can also apply regular expressions to check if the string represents such a pattern as with the preg_match() function and also check the length of the string with the strlen() function.
- Use the PDO object. It provides full immunity to SQL injections if you separate the data from the processing (the queries prepared with the query markers or named parameters).
- Do a two-step authentication. Instead of checking the login and the password in the same request, we split this treatment in two. First we ask for the login, and if this is correct we then ask for the password.

At the server configuration level :
- By modifying the php engine configuration file called php.ini. This is a text file that contains a set of directives that allow you to customise certain aspects of the PHP engine's behaviour. The magic_quotes_gpc directive can be used to automatically escape special characters in incoming strings (which come from forms, URL bars and cookies). It actually acts like the addslashes() function but in an automated way. -> magic_quotes_gpc = On

https://www.softwaresecured.com/introduction-to-sql-injection-mitigation/
https://www.chiny.me/l-injection-sql-14-2.php